CP/68

An M6800 Operating System

by **Jack E. Hemenway**

and

Robert D. Grappel

Edited by Edward R. Teja

Hemenway Associates, Inc.
101 Tremont St. Suite 208
Boston, MA 02108
(617) 426-1931

ISBN 0-935026-00-2

The authors of the programs provided with this book have carefully reviewed them to ensure their performance in accordance with the specifications described in the book. Neither the authors nor Hemenway Associates, Inc., however, makes any warranties whatever concerning the programs, nor assumes responsibility of any kind for errors in the programs or for the consequences of any such errors.

Library of Congress Catalog Card #79-89895

Printed in the United States of America

Table of Contents

Part 1 2
General Information 2
Command Structure 2
System Commands 3
System Device Errors 14
System Error Messages 14

Part 2 17
Advanced User's Guide 17
Filename formatting 38
Directory handling routines 40
Disk-file sequential I/O 41
Initialization and warmstart 42
Deleting a file 42
Program chaining 43
User entries 43
Format of binary files 43
Examples of CP/68 usage 45

Part 3 52
Description of Routines 52
Resident 53
BIOS 53
Logical device handlers 55
Command-line interpreter 58
Command Processing routines 63
Transient Commands 71
ASSIGN 71
BOOT 72
DELETE 73
INITIALIZE 73
LINK 74
PIP 75
Character-oriented device handlers 75
SECURITY 78
SET 78
STATUS 78

Part 4 80
FORMAT Utility 80

Table of Contents

Part 5..... 82
Random files..... 82
What are random-access files?..... 82
Physical and logical records..... 82
Entry points..... 83
File-control block..... 83
Data structures..... 85
File routines..... 85
Error codes..... 87
Notes and warnings..... 89
Example..... 90
STRUBAL+ support..... 95
Using files in STRUBAL+..... 97
Deleting files..... 98

Part 6..... 99
Modifications..... 99
Disk-handling software..... 101
Modifications for monitor ROMs..... 101

Part 7..... 102
Software listings..... 103

Introduction

CP/68 is a floppy-disc-based operating system that supports standard peripherals such as a line printer, CRT console, paper-tape reader and punch, and auxiliary consoles. The preliminary specification was described in EDN's Software Systems Design Course (Chapter 7), November 20, 1978. The current version of CP/68 is based on that specification and an improvement on it.

The operating system's modularity makes it easy to manage conceptually, and a pleasure to use. It is the most powerful system available for the 6800 family of microprocessors.

This book presents the entire operating system in seven distinct parts. Part I introduces you to the operation of the program; Part II adds the Advanced User's Guide; Part III covers the system's operation in detail; Part 4 explores the operation of the formatting utility; Part 5 introduces the random-access file support; Part 6 provides the information you will need to adapt the software to nonstandard hardware configurations; Part 7 gives complete source listings.

CP/68 GENERAL INFORMATION

COMMAND STRUCTURE

CP/68 commands consist of a command name and optional parameters. Some commands are memory resident and will execute immediately; others are transient (stored on disk) and must be loaded from disk before they are executed. System command names may be abbreviated to three characters; user-defined commands are invoked by entering their full names. These command files must be binary type with transfer addresses (type 01).

Where CP/68 requires numeric values, either decimal or hexadecimal notation may be used. Hex values must be preceded by a dollar sign ($). The period (.) is used for an operator prompt.

FILENAMES

File names in CP/68 consist of two parts: a name and an extension. The name is a string of alphanumeric characters up to 8 characters in length. The extension consists of up to 3 alphanumeric characters. The first character of both the filename and extension MUST be an alpha character. The name is separated from its extension by a period. The following are valid filenames:

INPUT.TXT MYFILE.B H1.HEX JACKS.FIL

To specify a file, give the disk drive number, filename and extension. The drive number is given as a decimal digit followed by a colon. The following are examples of unique files:

0:INPUT.TXT 1:INPUT.TXT 1:INPUT.HEX 0:INPUT2.TXT

If the drive number is zero it may be omitted. The following identify the same file:

0:BOBS.BIN BOBS.BIN

Note that only alphanumeric characters may appear in filenames or extensions. The following are invalid filenames:


```
1:JACKSFILE.HEX      (name more than 8 characters)
2:TEMP.FILE          (extension more than 3 characters)
0 TEST.TMP           (colon missing after drive number)
STBL+.BIN            (+ is a non-alphanumeric character)
EDITOR               (file extension missing)
```

WILDCARD FEATURES

CP/68 permits manipulation of classes of files. The mechanism for forming such classes is called wildcarding. Two wildcard characters perform unique identification tasks. The asterisk (*) matches an entire string of characters of arbitrary length. Since a complete filename consists of two strings, a name and an extension, the wildcard filename *.* is a short form of expressing all possible filenames. The wildcard filename *.HEX expresses all filenames with the extension HEX.

The second wildcard character is the question mark (?). This character substitutes for any single character, including a blank. Hence, the filename TEST?.HEX is equivalent to TEST.HEX or TESTP.HEX or TEST2.HEX. It is not equivalent to TESTING.HEX. The filename *.* is equivalent to ????????.???.

CP/68 SYSTEM COMMANDS

ASSIGN (transient)

This command assigns logical device names to physical devices. CP/68 supports the following logical devices.

CON	the console terminal I/O device
PTR	paper-tape reader
PTP	paper-tape punch
DSK	disk drive
LPT	line printer
MTA	magnetic tape
TTY	teletype (could be second console with paper-tape facilities)
NUL	null device (bit bucket)

ASSIGN manipulates the relationship of physical devices to logical device names. For instance, if it is desired to use the teletype as the console device, you need only enter

```
ASSIGN CON=TTY
```

CAUTION: Take care with assignments. It is possible to get into trouble. The console device should ALWAYS be a device capable of input.

Now, all console messages and input will use the teletype physical device. Suppose, however, that one wanted to test a routine which would simply output characters. The following command could be used to direct the paper-tape punch output to the Null device:

```
ASSIGN PTP=NUL
```

Devices can be repeatedly ASSIGNed. The STATUS command will give the present state of the device assignments.

```
ASSIGN CON=LPT
```

would lock up the system requiring a restart. One should not re-assign the DSK device, as that is where the system gets its transient commands.

The command can make as many assignments as desired at one time. After each command line, it will re-prompt for another command line. Enter the escape character followed by a CR (see SET under ES for a definition of this input), to leave ASSIGN and return to the command level.

BOOT (transient)

When a fresh copy of the system file is brought into the transient area from the disk, the system is said to have been booted. Any file which was LINKed on the disk in drive zero can be BOOTed. BOOT works as a specialized LOAD.

DELETE (transient)

This command is used to remove a file from disk. Wildcard characters in filenames can be used to remove categories of files. DELETE can process multiple command lines.

```
DELETE [drive:] filename.ext
```

where the drive number will default to drive zero. The filename and extension fields may contain wildcard characters. When the named file(s) are found the command issues a prompt that gives the user a chance to save the file.

```
DELETE  MYFILE.TMP

DELETE-0:MYFILE.TMP ? .YES
```

The YES response assures the operating system that this is the file

to be deleted. The YES can be abbreviated to Y; any input other than Y is interpreted as a NO.

```
DELETE *.TMP

DELETE-0:MYFILE.TMP ? .NO
```

would be the correct response if MYFILE.TMP was not the one that was to be deleted. This strategy saves you from being wiped out by typos. If there are several matches--due to the use of a wildcard character in the filename--each will be prompted in turn, and any of the matches may be removed. Suppose, for example, that there are three TXT files on drive 1, named TEST1, TEST2, and TEST3. Then the following command:

```
DELETE 1:TEST?.TXT

DELETE-1:TEST1.TXT ? .NO

DELETE-1:TEST2.TXT ? .YES

DELETE-1:TEST3.TXT ? .YES

DELETE .
```

This removes files TEXT2.TXT and TEXT3.TXT while leaving TEXT1.TXT . Enter the escape character followed by a carriage return to leave DELETE and return to the command level.

DIRECTORY (resident)

The DIRECTORY command provides a list of the files on a specified disk. The listing prints on the console device unless directed to the line printer.

```
DIRECTORY           (goes to console)

DIRECTORY /L        (goes to lineprinter)
```

The directory listing has the following format:

```
NAME TYPE ACCESS FIRST-TRACK/SECTOR LAST-TRACK/SECTOR SECTOR COUNT
```

The type code, access code, track/sector, etc. are output in hexadecimal. The type codes defined in CP/68 are:

```
00     binary file
01     binary file with transfer address
02     random access
03     text file (hex file)
```

The access codes defined in CP/68 are:

00 can be renamed or deleted
01 can be renamed but not deleted
02 cannot be renamed or deleted

Filenames are listed as 8-character strings with 3-character extensions. Following the directory list, the total number of disk sectors used by the listed files is listed in decimal.

The DIRECTORY command allows several levels of file qualification for listing categories of files.

DIRECTORY [/L] [drive:] [filename.ext]

The drive will default to disk zero. The filename and extension may use wildcards. For example:

DIRECTORY 1:*.HEX

will list on the console all files from disk 1 which have the extension HEX . Another example:

DIRECTORY /L TEST?.*

will list on the line-printer all files from disk zero which have names beginning with TEST followed by a character (or blank).

EXIT (resident)

This command causes control to shift from CP/68 to the underlying hardware system monitor. To get back to CP/68 either jump to the cold-start location or re-boot the system.

INITIALIZE (transient)

The INITIALIZE command formats a specified disk. All disks must be initialized before they can be used with CP/68.

INITIALIZE drive number

The drive number must be present even if the drive is zero. The command echoes the drive number, allowing the user to save the disk's contents.

```
INITIALIZE 1
INIT. DISK IN DRIVE 1 ? .YES
```

will begin the initialization process on the disk in drive 1. The initialization process wipes out the entire contents of the disk.

```
INITIALIZE 2
INIT. DISK IN DRIVE 2 ? .NO
```


returns to the command level leaving the disk unchanged. Upon completion of the initialization process (which may take several minutes) CP/68 returns to the command level.

JUMP (resident)

This command allows the user to leave CP/68 and go to any arbitrary absolute address. If the program at that address does a subroutine return (RTS instruction), CP/68 will continue at the command level.

```
JUMP $E113
```

will go to the address E113 (hexadecimal).

```
JUMP 256
```

will go to the address 256 (decimal).

LINK (transient)

This command sets the linkages for BOOT; it prompts the user for a file name. This file must be a binary file with transfer address (type 01). Once performed, the file named in LINK will be the file BOOTed when that disk is in drive zero.

```
LINK

ENTER SYSTEM FILE ? .CP68.SYS
```

links the file CP68.SYS as the file to be bootstrapped. The drive number defaults to zero; no wildcard characters are permitted in the filename or extension.

LOAD (resident)

This command puts programs into the transient area. They are not executed; control returns to CP/68 command level. LOAD requires that files be binary type (00 or 01 type).

```
LOAD [drive:] filename.ext
```

where the drive will default to zero. No wildcard characters are permitted in the filename or extension.

```
LOAD 1:PROG1.BIN
```

loads PROG1.BIN into the transient area.

PIP (transient)

The peripheral-interchange program (PIP) provides the operations for media conversion (eg, load, print, copy and combine disk files), referring to each peripheral device by name.

PIP destination=source[,source][,source].....

where destination and source are:

[drive:] filename.ext
device

Device is one one of the logical devices (see ASSIGN).

In the case of a disk-to-disk copy, for example,

PIP newdrive:=source drive:

copies the contents of the source drive exactly (sector for sector) onto the disk in newdrive. PIP prompts the user, providing a chance to save the contents of the newdrive.

PIP 0:=1:

COPY FROM DRIVE 1 TO DRIVE 0 ? .YES

will make the disk in drive 0 an exact copy of the disk in drive 1. A selective disk-to-disk copy follows a different form.

PIP destination:=source:filename.ext

where the filename and extension may contain wildcards. This will cause copies of all files on the source disk which match the filename and extension to be reproduced on the destination disk. All files on the destination disk are untouched; only those new files which were copied from the "source" disk will be written on the destination disk. If files with the same filename.ext already exist on the destination disk, an error indication is printed and the file is not copied.

PIP 0:=1:*.REL

copies all files on drive 1 that have the extension REL onto drive 0. The following command will copy all files from disk 0 to disk 1.

PIP 1:=0:*.*

This is a different form of copy from PIP 1:=0: . Using the wildcard filename.ext will copy the files into as sequential as possible a format on the new disk. Only the data sectors are copied, not the entire disk. Also, this form prompts the user as each file is copied, allowing very selective copying.

PIP can also transfer data between devices. For example, the following command can be used to view the contents of a file containing ASCII text:

```
PIP CON=filename.ext
```

Similarly, the contents of the file could be printed using PIP.

```
PIP LPT=filename.ext
```

PIP can be used to create text files.

```
PIP filename.ext=CON
```

builds a new file with the data typed at the console device. The END-FILE character (control-D, hex 04) is used to end the file. PIP can be used to transfer data from device to device as follows:

```
PIP LPT=CON          (print data from console)
PIP PTP=PTR          (duplicate a paper-tape)
PIP TTY=CON          (type from one device to another)
```

and many other combinations. PIP allows the user to combine several sources of input into one. This can be used to append several files into one, as in:

```
PIP bigfile=file 1, file 2, file 3, ....
```

Input from several devices can also be combined.

```
PIP newfile=oldfile,CON
```

lets you type new data after the oldfile is copied to the newfile.

PIP can also perform data translations. Internal storage of programs is in a binary format which cannot be listed, printed or copied to ASCII-character devices. PIP can convert the internal binary format to a hexadecimal format (MIKBUG) which can be used for listing, etc. Such data can also be converted into binary format. The following command converts a MIKBUG paper-tape file into an internal hexadecimal file:

```
PIP MYFILE.HEX/H=PTP
```

The following command can be used to convert the hex file into an internal binary file:

```
PIP MYFILE.BIN/B=MYFILE.HEX
```

PIP can be used to punch MIKBUG format tapes as follows:

```
PIP PTP/H=filename.BIN
```

so that, for example, one could punch a copy of the system with the

command:

```
PIP PTP/H=CP68.SYS
```

where CP68.SYS is a binary file. PIP can also be used to list or view a program file as follows:

```
PIP LPT/H=INIT.CMD   (transient INITIALIZE )

PIP CON/H=CP68.SYS
```

When copying a file from one disk to another, PIP maintains the filetype, and sets the access code to 00. It may be desirable at times to force the type of a file to TEXT (03). This can be done as follows:

```
PIP 1:TEMP.TXT/T=CON
```

The switch /T makes 1:TEMP.TXT a text file. Otherwise, a file produced by PIP will default to binary type. (00)

PIP can process multiple command lines. It will prompt the user after each command is completed. Enter an escape character to return to command level in CP/68.

RENAME (resident)

To change the name of a file without modifying its contents, use

```
RENAME [drive:] oldname.oldext,newname.newext
```

where the drive will default to zero. The file access code must be 00 or 01 to allow renaming. The newname must not exist already with that extension. The following command, for example, will rename the file BOBS.OLD to BOBS.NEW:

```
RENAME BOBS.OLD,BOBS.NEW
```

No wildcard characters are permitted in either the new or old names or extensions.

SAVE (resident)

This command saves an area of memory as a binary file.

```
SAVE [drive:] filename.ext,startad,endad [,transfer ad]
```

where the drive defaults to zero. The filetype of the save-file will be 00 if no transfer address is present, and 01 if a transfer address is supplied. For example, the following command will save the first 8k of memory as a system file to be entered at the address 07BC hexadecimal.

SAVE 1:CP68.SYS,$0000,$2000,$07BC

Addresses can also be entered in decimal notation. To save the first 256 bytes of memory:

SAVE BASEPAGE.SAV,0,256

No wildcard characters are permitted in the filename or the extension.

SECURITY (transient)

The files's security is determined by its access code. (see DIRECTORY). The code permits protection of certain files from deletion or renaming. SECURITY loads into the transient area. Its syntax is

SECURITY [drive:] filename.ext,access-code

where the drive will default to zero. For example, to remove any protection from the file CP68.SYS on drive zero:

SECURITY CP68.SYS,0

or to protect the file INIT.CMD from deletion:

SECURITY INIT.CMD,2

To allow INIT.CMD to be renamed but not deleted:

SECURITY INIT.CMD,1

No wildcard characters are permitted in either the filename or extension.

SET (transient)

This command allows the user to control the characteristics of the console and lineprinter devices.

SET parameter=value

where the following parameters are defined for the console:

BS -- Backspace character. This character may be set to any ASCII character on the console device. Control-H (08H) is the default.

DL -- delete character; causes the entire line just entered to be deleted. Control-U (15H) is the default.

DP -- depth count. The console will be paged with DP lines per page. This can be used to avoid scrolling; defaults to zero which disables paging.

WD -- Width. Sets the number of characters that will appear on a line. The default (zero) disables the line limit.

NL -- null count. Sets the number of nonprinting null characters sent with each carriage return. Allows delays for mechanical terminals. The default is zero.

TB -- tab character. Defines the character to be decoded as a tab. Default is Control-I (09H).

DX -- duplex switch. Selects either full or half duplex operation for the console. Default is F (full); H is half duplex.

EJ -- eject count. The number of lines skipped at the end of each page. If the pause switch is set the system waits for an escape character before continuing. Defaults to zero.

ES -- escape character. Defines the escape character; default is the ASCII escape character (1BH).

PS -- pause switch. Determines whether or not the system will wait at the end of a page. Valid values are Y (yes) and N (no); default is N.

Two parameters are exclusively for the line printer.

LD -- depth. Sets the number of lines per page; defaults to 60 decimal.

LW -- width. Sets the number of characters per line; defaults to 80 decimal.

With the exception of DX and PS, all parameters take a number which may be either decimal or hexadecimal. The following are some valid commands:

```
SET LD=50          (50 lines per page on LPT)
SET DX=H           (half-duplex CON)
SET BS=$08         (backspace CNTL-H)
SET EJ=0           (no formfeeds)
SET PS=Y           (pause on)
```

SET allows multiple command lines. It will prompt after each command line. Enter an escape character followed by a carriage return to return to command level.

STATUS (transient)

A systems status is its list of the present state of device assignments--printed on the current console device. It returns directly to command level after listing the devices. See ASSIGN for a complete list of device names.

SUBMIT (resident)

This command allows the use of a file containing CP/68 command lines as a source of console commands. The text lines in the file are used as though they were typed at the console. The memory resident SUBMIT can invoke any other command under CP/68. The file must be a text file (type 03), built with either the editor or PIP. The syntax of the SUBMIT command is:

SUBMIT [drive:] filename.ext

where the drive defaults to zero. All commands from the file will be echoed as they are read. There is a special divert character used in SUBMIT files. This is the ampersand "&" symbol.

The use of the divert character allows a one-line console command to be inserted into a SUBMIT command string. When "&" is found in a SUBMIT file, the user is prompted for a command. This command is executed, and then the SUBMIT file is resumed. When the end of the file is encountered, the system returns to command level at the console. For example, suppose the file SUBMIT.TXT contains the following:

```
DIRECTORY 1
STATUS
ASSIGN PTR=TTY
(escape)
&
LOAD INIT.CMD

end of file
```

Then, the following command:

SUBMIT SUBMIT.TXT

would first list the directory of drive 1, give the device status of the system, assign the PTR device to the TTY, escape to command level, accept a user command from the console and execute it, load the file INIT.CMD, and return to command level.

No wildcard characters are permitted in the filename or extension.

SYSTEM DEVICE ERRORS

All device errors in CP/68 are reported in the following format:

device-name ERROR: number

where device-name is the three-character logical name and the error number is hex encoded. For example:

```
LPT ERROR: 0A
DSK ERROR: 02
```

are system device error messages. The set of errors defined in CP/68 are:

- 01- end of directory found in search
- 02- file already in use
- 03- file already exists
- 04- no such file exists
- 05- read/write error
- 06- directory overflow
- 07- disk full
- 08- end-file encountered
- 09- bad disk sector, bad media
- 0A- device not ready
- 0D- illegal use of File Control Block
- 12- illegal operation (write a read file, etc.)
- 15- bad file name

CP/68 SYSTEM ERROR MESSAGES

FORMAT ERROR — The command line does not conform to the syntax specified for the command.

NUMBER ERROR — A bad numeric argument is present. The drive number is out of range or is not followed by a colon.

FILE NOT FOUND — The requested file could not be found.

DISK ERROR:aa AT SECTOR bb, TRACK cc — This error message comes from the INITIALIZE command. The error type (aa) is a device-error number.

SYNTAX ERROR
INVALID SET PARM

These error messages come from the SET command. They indicate a bad SET command line.

The following errors come from PIP:

BAD INPUT (OUTPUT)
A device error; usually accompanied by a device-error message.

ILLEGAL INPUT (OUTPUT) DEVICE
Refers to attempts to use a device in an invalid manner, such as reading from a lineprinter.

BUFFER OVER-RUN
An overly long input line was encountered. The input file is probably the wrong type for the operation desired.

ILLEGAL SWITCH
Indicates a syntax error in the switch portion of the command line.

READ (WRITE) ERROR
Encountered in disk-to-disk copying; accompanied by a device error message.

DIRECTORY ERROR
The directory on a disk could not be read properly. This message is usually accompanied by a device-error message.

CHECKSUM ERROR
The checksum of a hex-formatted file was not correctly read.

Additional errors are:

SUBMIT FILE ERROR — The filename in the SUBMIT command line could not be found or was not a TEXT file.

ILLEGAL FILE TYPE — The file specified for LOAD was not a binary file.

RENAMING ERROR
DUPLICATE NAME
SECURITY ERROR

These errors messages come from the RENAME command. RENAMING ERROR indicates some form of disk error in accessing the drive containing the old file. DUPLICATE NAME indicates that the new name already exists on the disk. SECURITY ERROR indicates that the old file is protected from renaming. (access code=01 or 02)

UNABLE TO CHAIN: filename.ext

This error message indicates that a CHAIN request was made to the CP/68 system with filename.ext but it cannot be done. (no such file, disk read error, file not the right type, etc.)

FILE DELETE-PROTECTED

This file is protected from deletion (access code 02). It cannot be deleted until its access code is reduced.

DELETE ERROR-OPEN OUTPUT FILES

As long as any output files are open CP/68 cannot delete a file on that disk.

Part 2

Advanced User's Guide
****

INTRODUCTION

CP/68 is fully relocatable, supports dynamic disk files on multiple drives, has a clean and logical command structure, provides device-independent I/O, and has features which facilitate complex system operations. It requires slightly less than 8K bytes of contiguous memory plus a section of base-page (0020H to 0046H). Transient files overlay some system commands and user files. User files can chain in new files. Files can be used as a source of system commands.

CP/68 provides an extensive set of "extended instructions" which greatly add to the power of the 6800 instruction set. These "extended instructions" were used frequently in CP/68 itself. This portion of the book describes the structures and algorithms used in CP/68 in sufficient detail to allow you to add functions to the system and to interface your own programs to CP/68.

CP/68 uses several data structures in memory to perform various functions. These data structures are involved in all I/O operations, and some of the other system operations. The data structures discussed in this section include: Base-page, Equipment table, Physical device table, Request-control block (RCB), File-control block (FCB), File information block (FIB), and stack.

BASE-PAGE

CP/68 uses an area of base-page memory from address 0020H to 0047H to store global variables and system parameters. Most of these locations deal with I/O, while others are involved with command parsing and other functions.

Command-parsing variables

DESCRA 0020H

This 2-byte location stores the address in memory of the beginning character of a token. (For a description of "tokens", see the CP/68 operation NXTOK)

DESCRC 0022H

This byte stores the number of characters in the current token.

CUCHAR 0023H

This 2-byte location stores the address of the next character in the command line to be processed. Typically, this means that CUCHAR=DESCRA+DESCRC+1. CUCHAR is initialized to the beginning of the command line when it is desired to parse that line. DESCRA is automatically set by the NXTOK operation. To back up a token, set CUCHAR=DESCRA.

RC 0025H

This byte returns the return-code of the extracted token. (See NXTOK for a description of token codes.)

CLASS 0026H

This byte returns the class of the extracted token. The class is a sub-classification of the RC. (See NXTOK for a description of token classes.)

VALUE 0027H

This 2-byte location stores the binary value of a numeric token when one is encountered during parsing. It is an unsigned 16-bit number.

Conversion from hex or decimal bases is done automatically by CP/68.

Disk information locations

FCBCHN 0029H

This 2-byte location stores the address of the header of the linked list of open file-control blocks. If FCBCHN is zero, there are no open files. If FCBCHN is not zero, it contains the address in memory of the first FCB that is active. Each FCB contains a pointer to the next FCB. If the pointer is zero, the end of the chain has been reached.

FRETAB 002BH

This is a table consisting of four, 2-byte entries. Each entry corresponds to one of the four disk drives maintained by CP/68. The entry stores the track and sector numbers of the header of the free-space chain on that disk. When a disk is being used, CP/68 copies the header data into the FRETAB entry so that it does not have to continually read the data from the disk. The entries are cleared when CP/68 is re-started.

Unused locations

0033H to 0039H Reserved for future expansion.

Console parameters

BS 0039H

This byte is the character to be used as a backspace on the console device. The default value for BS is 08 hex.

DL 003AH

This byte is the character to be used as the line-delete on the console device. The default value for DL is 15 hex (control-U).

DP 003BH

This byte is the number of lines per page on the console device. The default value for DP is 00 hex. (no limit on page depth)

DPCNT 003CH

This byte is used as the counter for the lines on a page on the console device. When DPCNT=DP, the end of a page has been reached. DPCNT is initialized to 01 hex.

WD 003DH

This byte is the number of characters per line on the console device. The default value for WD is 00 hex. (no limit on line width)

NL 003EH

This byte is the number of nulls which will be output with each linefeed on the console device. This feature allows linefeed delays for consoles which need such delays. The default value for NL is 00 hex.

TB 003FH

This byte is the character to be recognized as a tab on the console device. The default value for TB is 09 hex. (control-I)

DX 0040H

This byte is a switch which determines if the console device is to echo input characters. (Full or half duplex) If DX=00, the console is full-duplex and will echo all input. If DX=FF, the console is half-duplex and will not echo. The default value for DX is 00 hex. (full duplex)

EJ 0041H

This byte is the number of linefeeds to be output at the end of a page on the console device. The default value for EJ is 00 hex.

PS 0042H

This byte is a switch which controls the "pause" feature on console output. If PS=00, the console will wait at the end of a page of output until an escape character is input. (See ES below) If PS is not zero, the console will not pause. The default value of PS is FF hex. (no pause)

ES 0043H

This byte is the character to be interpreted as an "escape" on console input. The default value for ES is 1B hex. (ASCII "ESC")

LDP 0044H

This byte sets the number of lines per page on the lineprinter device. The default value for LDP is 60 decimal.

LDPCNT 0045H

This byte stores the count of lines on a page of lineprinter output. When LDPCNT=LDP, a full page has been output. The value of LDPCNT is initialized to 00 hex.

LWD 0046H

This byte sets the number of characters on a line for the lineprinter device. The default value for LWD is 80 decimal.

EQUIPMENT TABLE (EQTAB)

The Equipment table, in conjunction with the Physical-device table, vectors I/O using the device name provided by the user in the RCB or FCB. Each table contains an entry for each physical device in CP/68. The physical devices are: Console (CON), Papertape reader (PTR), Papertape punch (PTP), Disk (DSK), Lineprinter (LPT), Magnetic tape (MTA), Teletype or alternate Console (TTY), and Null device (NUL). The CON device is the command source. It must be capable of input and output of ASCII characters. The CON "SET" parameters control its behavior. The PTR device is input only. The X-ON (11 hex) and X-OFF (13 hex) characters are used to turn PTR on and off. Linefeeds (0A hex) and nulls (00 hex) are swallowed. The PTP device is output only. A linefeed (0A hex) is issued with each carriage return (0D hex) and 4 nulls (00 hex) are added. The DSK device is a floppy-disk drive. The details of its operation are handled in the system code. The LPT device is an output-only printer. The LPT "SET" parameters control its behavior. The formfeed (0C hex) character is used to control paging on the LPT device. Linefeeds (0A hex) are automatically provided with each carriage return. (0D hex) The MTA device is unsupported in the present CP/68. The NUL device is actually not a device at all but simply a "bit bucket" or "do nothing". This proves useful at times to check out programs. Each device is given a three-character name.

Each entry in the equipment table has three 2-byte fields. The first field is the address of an input routine for that device. This routine must handle a line or block of data; CP/68 does not use character or single-byte I/O. If the device does not support input (the LPT for example), then the NUL handler is used. The second field is the address of an output routine for that device. This routine must also handle a

line or block of data. If the device does not support output (the PTR for example), the the NUL handler is used. The third field is the address of the interface used by that device.
As supplied, CP/68 assumes the following:

```
CON     ACIA at 8008H
PTR     ACIA at 8010H
PTP     ACIA at 8010H
DSK     special case...the handlers for this device have
        interface addressing built in.
LPT     PIA at  8002H
MTA     not implemented in the current version.
TTY     ACIA at 8010H
NUL     no device needed
```

Note that the PTR,PTP, and TTY devices are set up to share one interface. This allows using the papertape facilities of a teletype (ASR-33) as well as its keyboard/printer. Note that CP/68 initializes the CONsole ACIA device, the TTY ACIA device, and the LPT PIA device on cold start. Other devices will need to be initialized by the user. An example Equipment table entry is shown below.

```
CONSOL FDB INLIN      input a line from the console
       FDB OTLIN      output a line to the console
       FDB $8008      ACIA at address 8008H
```

PHYSICAL DEVICE TABLE (PDTAB)

This table vectors I/O calls to the proper entry of the equipment table. Each entry in this table consists of three fields. The first field is the three-character name of the device; the second field is the address of the entry in the equipment table which services the physical device; the third field is also the address of the equipment table entry. The use of both fields allows for reassignment of a physical device. Suppose, for example, that you wanted to use the TTY device as the console. (See the ASSIGN command) You would modify the second entry of the physical-device table CON entry to point to the TTY entry of the Equipment table. All I/O directed to CON would then be vectored to the TTY device using the TTY handlers. The third field of the physical-device table entry is used to maintain a pointer to the original address of the device. Thus, no matter how many times a device may have been reassigned, there is still a pointer to its original Equipment-table entry. This is needed by some CP/68 commands, such as STATUS. Hence, each entry in the physical-device table has seven bytes. As an example, here is the CON entry.

```
FCC 'CON'     name is CON
FDB CONSOL    Equipment table pointer
FDB CONSOL    "same"
```

The physical-device table uses a zero entry as an end marker.

REQUEST-CONTROL BLOCK (RCB)

All requests for I/O through CP/68 require a data structure in memory called an RCB or FCB. An RCB consists of 9 bytes of memory. Disk I/O requires the extended block (FCB). All other I/O requests may use an RCB. There are five fields in an RCB; three must be filled in by the user and the system provides the other two. The structure of an RCB is as follows:

RCBEQT supplied by the system

This 2-byte space is the address of the EQTAB entry which applies to this request for I/O.

RCBGDT required from user

This three-byte space must contain the three-character name of the device from-or-to which I/O is desired. CP/68 looks up this name in PDTAB and uses the entry there to find the EQTAB entry which it stores in RCBEQT.

RCBSTA supplied by the system

This byte is the status of the I/O request. It should be cleared before a CP/68 I/O request is issued. It returns any error conditions. It is zero for successful I/O completion. If RCBSTA returns nonzero, an error has occured.

RCBDTT required from user

This byte is a switch to choose input or output. If RCBDTT=0, then input is being requested. If RCBDTT=FF, then output is being requested.

RCBDBA required from user

This 2-byte space should contain the address in memory of a buffer to be used for I/O. It is up to the user to provide sufficient space in the buffer.

Example of RCB setup for CONsole input

```
        RMB 2           space for RCBEQT
        FCC 'CON'       RCBGDT
        FCB 0           RCBSTA
        FCB 0           RCBDTT input
        FDB BUFFER      buffer address
```

Example of RCB setup for PTP output

```
        RMB 2           space for RCBEQT
        FCC 'PTP'       RCBGDT
        FCB 0           RCBSTA
        FCB $FF         RCBDTT output
        FDB BUFFER      buffer address
```

To access fields in the RCB, the following EQUates will be useful.

```
        RCBEQT EQU 0
        RCBGDT EQU 2
        RCBSTA EQU 5
        RCBDTT EQU 6
        RCBDBA EQU 7
```

Now, if the index register points to the RCB address...

```
        LDA A RCBSTA,X      get the status
        LDX   RCBDBA,X      get the buffer address
```

and so on.

FILE-CONTROL BLOCK (FCB)

This data structure is an extended RCB with additional fields necessary for disk I/O. It consists of 42 bytes of memory. The first five fields are identical to the RCB fields.

```
      FCBEQT=RCBEQT
      FCBGDT=RCBGDT
      FCBSTA=RCBSTA
      FCBDTT=RCBDTT
      FCBDBA=RCBDBA
```

There are 14 additional fields in an FCB.

FCBDRV required from user

This byte must contain the drive number of the disk containing the desired file. Drive numbers run from 0 upwards.

FCBTRK supplied by system

This byte must contain the track number of the desired sector on the disk in FCBDRV.

FCBSCT supplied by system

This byte must contain the sector number desired on FCBTRK.

FCBFWD supplied by system

This 2-byte space is filled in by CP/68 with the forward link (track and sector) of the requested sector in disk reads and writes.

FCBBAK supplied by system

This 2-byte space is filled in by CP/68 with the backward link (track and sector) of the requested sector in disk reads and writes.

FCBNAM required from user

This 13-byte field must contain the file name and extension of the desired file for use by the file-manager of CP/68. The file name must be exactly 8 characters; pad with blanks as necessary to fill 8 characters. The ninth character must be a period. "." The extension must be exactly three characters; pad with blanks as necessary to fill 3 characters. The 13th character should be an "end-string" character. (04 hex) A system function is provided to format a string of characters into this internal form...see FMTS.

FCBTYP user supplied for new file, system supplied for existing file

This byte gives the type of file. If a new file is being created, the user should set this byte as follows:

00 binary file
01 binary file with transfer address (runable)
02 random file
03 text or hex file

Other numbers may be used, but CP/68 type-checks files that are loaded into memory, copied, etc. If the file already exists, the file manager will fill this field with the file type.

FCBACS user supplied for new file, system supplied for existing file

This byte gives the access code of the file. If a new file is being created, the user should set the byte as follows:

00 no protection
01 file can be renamed but not deleted
02 file can neither be renamed or deleted

If the file already exists, the file manager will fill this byte with the access code of the file.

FCBFTS supplied by system

This 2-byte field is filled by the system with the first track and sector

of the named file.

FCBLTS supplied by system

This 2-byte field is filled by the system with the last track and sector of the named file

FCBNMS supplied by system

This 2-byte field is filled by the system with the number of sectors used by the named file.

FCBNFB supplied by system

This 2-byte field is filled by the system with a link to the next active FCB in the system. If this is the most recent FCB in the system, the link will be zero. (See FCBCHN in base-page)

FCBIND supplied by system

This 2-byte field is filled by the system with a pointer to the buffer supplied at FCBDBA. This pointer indicates the present data byte in the buffer.

FCBSCF required from user

This byte is a switch to control space-compression in text files. If FCBSCF=0 then no space-compression is performed. If FCBSCF is nonzero, then all spaces within a file (20 hex) will be compressed as follows:

Any data byte =20 hex will be compressed. Spaces are replaced by the negative (2's-complement) of the number of sequential spaces. Hence, if the file contained the following 5 bytes of data:

41 20 20 20 41 'A A'

it would be compressed to read

41 FD 41

where FD=-3 .

When a file is read back with FCBSCF nonzero, spaces are re-inserted where necessary. Only files of ASCII text should be compressed.

Example of FCB setup to read file MYFILE.TXT on disk 1

```
        RMB 2          FCBEQT
        FCC 'DSK'      FCBGDT=DSK
        FCB 0          FCBSTA
        FCB 0          FCBDTT=input
        FDB BUFFER     sector buffer address
```

```
FCB 1          FCBDRV=1
RMB 1          FCBTRK
RMB 1          FCBSCT
RMB 2          FCBFWD
RMB 2          FCBBAK
FCC 'MYFILE  '
FCC '.'
FCC 'TXT'      FCBNAM
FCB $04
RMB 1          FCBTYP
RMB 1          FCBACS
RMB 2          FCBFTS
RMB 2          FCBLTS
RMB 2          FCBNMS
RMB 2          FCBNFB
RMB 2          FCBIND
FCB $FF        FCBSCF (compression on)
```

Here is a set of EQUates which will ease access of FCB fields.

```
FCBEQT EQU 0
FCBGDT EQU 2
FCBSTA EQU 5
FCBDTT EQU 6
FCBDBA EQU 7
FCBDRV EQU 9
FCBTRK EQU 10
FCBSCT EQU 11
FCBFWD EQU 12
FCBBAK EQU 14
FCBNAM EQU 16
FCBTYP EQU 29
FCBACS EQU 30
FCBFTS EQU 31
FCBLTS EQU 33
FCBNMS EQU 35
FCBNFB EQU 37
FCBIND EQU 39
FCBSCF EQU 41
```

Thus, if the index register points to the FCB address

```
LDA A FCBFWD,X       get forward link track
LDA B FCBFWD+1,X     get forward link sector
STA A FCBTRK,X       put into track
STA B FCBSCT,X       put into sector
```

and so on.

FILE-INFORMATION BLOCK (FIB)

This data block contains the information in the file directory on disk. Each file has a FIB, consisting of 32 bytes. In the present CP/68, only the first 20 bytes are used. The FIB fields match the FCB fields starting with FCBNAM and ending with FCBNMS.

```
FIBNAM=FCBNAM
FIBTYP=FCBTYP
FIBACS=FCBACS
FIBFTS=FCBFTS
FIBLTS=FCBLTS
FIBNMS=FCBNMS
```

The FIBNAM field is always maintained in the proper format. The following EQUates will ease the access of FIB fields.

```
FIBNAM EQU 0
FIBTYP EQU 13
FIBACS EQU 14
FIBFTS EQU 15
FIBLTS EQU 17
FIBNMS EQU 19
```

STACK

CP/68 contains its own stack in its RAM space. Cold or warm starts reset the stack pointer to the system stack location.

CP/68 provides a 256 byte stack which is quite ample. Since system calls are done via software interrupts, and the stack is used for parameter passage, a minimum of 100 bytes of stack is needed to run CP/68 successfully.

DO NOT UPSET THE CP/68 STACK POINTER!

A disk initialized for CP/68 (see INITIALIZE command) has some data structure written onto it which CP/68 uses to work with files. These data structures must be maintained or CP/68 may do unpredictable things to the disk. An uninitialized disk will not work with CP/68.

Track 0

The first track on the disk (track 0) is reserved for the system. The first sector (sector 1) is used for bootstrap space, system linkage, and the free-space header. If SECSIZ is the number of bytes per sector on the disk, then

SECSIZ-5	first track of system-linked file
SECSIZ-4	first sector of system-linked file
SECSIZ-3	last track of system-linked file
SECSIZ-2	last sector of system-linked file

(These values are written by the LINK command)

SECSIZ-1	track of first free sector
SECSIZ	sector of first free sector

(These values are initialized by INIT, updated by file manager.)

The beginning SECSIZ-6 bytes of the first sector of the first track provides space for a bootstrap program. The remainder of track 0 is space for the file directory information. Files are described by 32-byte FIB blocks that are stored sequentially as long as there is space. The directory space is initialized to all zero by INIT.

A directory search is terminated when a zero is found at the start of a FIB block. A FIB is removed from the directory by placing a blank in the first character of the file name field (first byte of FIB). This does not recover the file's sectors, however. The DELETE function is provided to both remove a FIB and replace the file's sectors on the free-space list of the disk. The next file to be created will use that space.

Tracks 1-n

The rest of the tracks on the disk are used as CP/68 file space. Every sector has forward and backward links in its first four bytes. These links are automatically maintained by the system. Hence, each sector has SECSIZ-4 usable bytes. An initialized disk has its sectors linked in a pattern found to optimize access times, not usually in a sequential manner. The free-space chain header on track 0 points to the start of this list; sectors are allocated to files from this list and links changed accordingly. Deleted files return their sectors to the head of the free-space list. A much-used disk will become "fragmented"--the links will be very far from sequential. This increases access times, but CP/68

will not lose data as long as the links are maintained. The PIP command provides a way to "compact" a disk that has become fragmented.

(Note: the backward links are not used in the present CP/68.)

ISSUING SUPERVISOR CALLS (SVC)

CP/68 was written to be relocatable. Each routine could not have an absolute address. Also, it was desired that routines have standardized calling sequences and that registers be saved in most cases. The mechanism of the 6800 software interrupt was used to solve the problem of calling CP/68 routines. CP/68 has only two entry points: the cold start at its first byte, and the software-interrupt handler (SWIHDR) three bytes later. All system calls are performed by a software interrupt (SWI) instruction followed by a routine number. These two bytes are collectively referred to as an SVC. CP/68 automatically vectors the call to the appropriate address. The SWI saves the registers on the stack and recovers them on return from the system. Those routines that use registers for parameters manipulate them on the stack. Once CP/68 has been called, the stack contains:

```
stack pointer:
              SWIHDR return address
              condition code byte
              accumulator B
              accumulator A
              Register X
              Return address
```

Thus, the following code would recover the contents of the B accumulator.

```
              TSX
              LDA B 3,X
```

The following would return the condition codes to the user.

```
              TPA
              TSX
              STA A 2,X
```

Since each CP/68 routine call is done in the same way, SWI and a byte, they can be made macros and used like new instructions. For example, CP/68 has a routine to read a byte from an open file. It would be called as follows:

```
              SWI      call CP/68
              FCB 24   file-read
```

A macro could be written:

```
READ  MACRO
      SWI
      FCB 24
      MEND
```

so that whenever a file read was desired, a READ instruction could be given. CP/68 was written with the express purpose of providing a list of useful "extended instructions".

Using the software-interrupt mechanism, up to 256 different system calls are possible. In fact, CP/68 uses only 54 of these. (numbered 0-53) An SWI followed by any number larger than 53 will be vectored to the usual SWI trap in the underlying monitor. (Check the SWIHDR routine for the location of this trap.) Thus, breakpointing can be done in CP/68 with a two-byte "SWI"

```
      SWI          call CP/68
      FCB $FF      force call to monitor
```

which will operate exactly like the simple SWI did without it. Programs that use SWI instructions must be modified to add the second byte, or CP/68 routines will be called with unpredictable results.

General instructions

00 PSHAL

This routine pushes all the register contents onto the stack in the normal 6800 order.

01 PULAL

This routine is the reverse of PSHAL. It restores the register contents from the stack.

02 TXAB

This routine transfers the contents of the index register to the A and B accumulators. The high byte goes into A, the low byte into B. The index register is undisturbed.

03 TABX

This routine is the reverse of TXAB. The contents of the A and B accumulators are transfered into the index register. The contents of A and B are not disturbed.

04 XABX

This routine exchanges the contents of the index register and the A and B accumulators. A and B become X, X becomes A and B.

05 PSHX

This routine pushes the contents of the index register onto the stack. The low byte is pushed first, followed by the high byte. No registers are disturbed.

06 PULX

This routine is the reverse of PSHX. The index register is loaded from the stack. Only the index register is changed.

07 ADXAB

This routine adds the 16-bit unsigned contents of the index register to the combined 16-bit value in the A and B accumulators. The result is left in A and B, X is unchanged. The condition codes are set to reflect the results of the addition.

08 ADABX

This routine works like ADXAB except that the result is left in X, A and B are unchanged. The condition codes reflect the results of the addition.

09 ADDAX

This routine adds the unsigned byte in the A accumulator to the 16-bit unsigned value in the X register. The result is in the X register, A is unchanged. The condition codes reflect the result of the addition.

10 ADDBX

This routine is like ADDAX except that the B accumulator is used. The condition codes reflect the results of the addition.

11 SBXAB

This routine subtracts the 16-bit unsigned value in the index register from the combined 16-bit value in the A and B accumulators. The result is left in A and B, X is unchanged. The condition codes are set to reflect the results of the subtraction.

12 SBABX

This routine is like SBXAB except that the result is left in X, A and B are unchanged. The condition codes reflect the results of the subtraction.

13 SUBAX

This routine subtracts the unsigned byte in the A accumulator from the 16-bit unsigned value in the index register. The result is left in the index register, A is unchanged. The condition codes are set to reflect the result of the subtraction.

14 SUBBX

This routine is like SUBAX except that the B accumulator is used. The condition codes reflect the results of the subtraction.

15 MUL8

This routine multiplies the unsigned bytes in A and B accumulators and puts the resulting 16-bit value high byte in A, low byte in B. The condition codes are set to reflect the product of the multiplication.

16 MUL16

This routine multiplies the unsigned 16-bit value in the index register by the 16-bit value in the A and B accumulators. The 32-bit result is left in A,B,X . The condition codes are set to reflect the result of the multiplication.

17 MOVC

This routine moves up to 256 bytes from one place to another. The from-address and to-address are placed on the stack. (to-address first, followed by from-address.) The byte count is passed in the B accumulator. On return, B=0, the stacked addresses have been incremented B times, and A is undisturbed.

```
Example:    LDX TOADDR     get to-address
            PSHX           use CP/68
            LDX FRMADDR    get from-address
            PSHX           use CP/68
            LDA B #100     move 100 bytes
            MOVC           move them
            INS
            INS            clean stack
            INS
            INS
```

18 CMPC

This routine compares two strings. It can be used for comparing text strings or other data. It can compare strings of up to 256 bytes in length. If the "end-string" character (04 hex) is found in either string, comparison is terminated. The parameter setup is the same as MOVC--the addresses of the two strings are stacked and the byte count goes into accumulator B. The result of the comparison is returned in the condition codes.

Example of using CMPC

```
            LDX #STRNG2    point to second string
            PSHX
            LDX #STRNG1    point to first string
            PSHX
            LDA B #10      compare 10 characters
            CMPC           compare
            INS
            INS            clean stack
            INS
            INS
            BGT ----       was string 1 > string 2?
```

so that if STRNG1='AAAAAAAAAAAAAAAAAAAAAAA' and if
STRNG2='BAAAA   ', then the branch would
not be taken.

45 MOVS

This routine works like MOVC except that it does not use a byte count in the B accumulator. The move continues until an "end-string" (04 hex) is found in the from-string.

46 INDEX

This routine adds the product of the unsigned bytes in the A and B accumulators to the 16-bit unsigned value in the index register. The result is left in the index register, A and B are unchanged. The condition codes are set to reflect the results of the operation.

50 DIV16

This routine divides the unsigned 16-bit value in the combination of the A and B accumulators by the 16-bit unsigned value in the index register. The quotient is returned in the A and B accumulators. The remainder is returned in the index register. The condition codes are set to reflect the quotient value.

Command-parsing routines

47 NXTOK

This routine breaks up a command line into "tokens". A token is a substring of the command line which is treated as a unit. CP/68 defines the following tokens:

NAME A name is a string of characters which begins with an alphabetic character and contains only alphanumeric characters. (no imbedded spaces)

NAME WITH WILD-CARD CHARACTERS

A name which may include the special characters "*" and "?".

NUMBER A string of digits which may be decimal or hexadecimal. Hexadecimal numbers must begin with a dollar sign. ($)

DELIMITER

One of the special characters defined by CP/68. This includes the period (.), comma (,), colon (:), dollar sign ($), equals sign (=), semicolon (;), and the arithmetic routines +,-,and /

CARRIAGE RETURN

The ASCII carriage return character. (0D hex)

ERROR A token not falling into one of the above classes.

NXTOK uses base-page for its parameters. Scanning the command line begins at the character whose address is in CUCHAR. The address of the first character of the token is returned in DESCRA. Note that spaces are not part of any token. Spaces are skipped over by NXTOK unless they are imbedded in a token. The count of the number of characters in a token is returned in DESCRC. The base-page locations RC and CLASS return the classification of the token as follows:

NAME	RC=01	CLASS=02
NAME (WCRD)	RC=02	CLASS=02
NUMBER	RC=03	CLASS=02
DELIMITER	RC=ASCII code of character	CLASS=04
CARRIAGE RET.	RC=0D hex	CLASS=0D hex
ERROR	RC=00	CLASS=00

CUCHAR is returned pointing one character beyond the end of the present token. If the token is a number (RC=03), then its binary value is returned in the base-page location VALUE. NXTOK will automatically convert unsigned decimal or hexadecimal numbers into binary form. The hex numbers must have a leading dollar sign. ($) NXTOK will trap numbers that are too large (>65535 or FFFF hex) as errors.

Example of use of NXTOK

command line='LOAD 1:MYFILE.EXT ' carriage return

```
first token='LOAD'     RC=01, CLASS=02
second token='1'       RC=03, CLASS=02, VALUE=0001
third token=':'        RC=3A, CLASS=04
fourth token='MYFILE'  RC=01, CLASS=02
fifth token='.'        RC=2E, CLASS=04
sixth token='EXT'      RC=01, CLASS=02
seventh token=c.r.     RC=0D, CLASS=0D
```

19 IOHDR

This is the basic I/O routine in CP/68. It is called with the address of the RCB or FCB in the index register and it causes the system to perform the I/O operation. No registers are disturbed by this routine. IOHDR handles entire lines or blocks of data at once. All CP/68 devices are handled through IOHDR, although some additional routines are provided for disk I/O and special cases of system I/O. The status of the I/O request is returned in RCBSTA (or FCBSTA).

Example of use of IOHDR to write character string on terminal

```
        LDX #RCB        point to RCB
        IOHDR
```

where the RCB has been set up as follows:

```
RCB     RMB 2           space for EQTAB
        FCC 'CON'       console device
        FCB 0           status
        FCB $FF         output
        FDB DATA        address of data characters

DATA    FCC 'THIS STUFF WILL BE PRINTED'
        FCB $0D         carriage return
```

Note that a carriage return was used to indicate the end of a line. CP/68 will add a linefeed automatically for CON, TTY, or LPT I/O. If a new line is not desired, use an "end-string" (04 hex) in place of the carriage return.

Reading or writing a disk sector is done through IOHDR by some additional setup in the FCB. The FCBGDT must be 'DSK'. The FCBSTA is cleared. The FCBDTT is set to 00 for reading or FF for writing. The FCBDBA is set to point to a sector buffer. The FCBDRV is set to the desired drive number. The FCBTRK is set to the desired track number. The FCBSCT is set to the desired sector number. IOHDR will perform the read or write to/from the indicated sector on the indicated disk. Any disk sector can be accessed in this manner. The only error checking performed is that the desired sector exists on the disk and that the desired operation can be performed by the drive. The user is warned that IOHDR does not preserve the links or other data structures on the disk. This is done by the routines READ, WRITE, etc.

30 PRTERR

This routine prints error messages for device I/O errors. It is called with the address of an RCB or FCB in the index register. If the status (RCB or FCBSTA) is zero (good), it does nothing. If the status is nonzero, it prints an error message on the console device. The error message is of the form:

AAA ERROR: BB

where AAA is the device name (RCB or FCBGDT) and BB is the status value (RCB or FCBSTA) in hexadecimal.

48 GTCMD

This routine accepts a command line from the console. The user is prompted and a new line may be typed in. GTCMD passes the line directly to NXTOK, so on return from GTCMD, the first token on the line has been parsed. If the user desires to back up to the start of the line, set CUCHAR=DESCRA in base page.

49 PRTMSG

This routine prints a string on the console device. The index register is pointed to the start of the string. If the string terminates with a carriage return, a new linefeed is issued. If the string terminates with 04 hex, no linefeed is issued.

Filename formatting

44 FMTFCB

This routine parses a complete file designation including drive number, filename, and extension, and places it properly formatted into an FCB. The format that FMTFCB expects is:

[drive:] filename.ext

where the drive number and colon are optional. If the drive number is omitted, drive 0 will be assumed. FMTFCB will allow no wild-card names; it works only with unambiguous file references. To use FMTFCB, place the address of the character string containing the file specification into CUCHAR in the base-page. Place the address of the FCB into the index register. FMTFCB will place the drive number into FCBDRV and the filename appropriately formatted into FCBNAM. Any error conditions are returned in FCBSTA. If FCBSTA=00, the file specification was correctly formatted. If there was some error, FMTFCB returns an error status=21 .

52 FMTS

This routine formats a filename from the input form which may vary in length to the fixed internal form. It also handles the expansion of wild-card characters. The calling sequence is like MOVC, with from and to addresses on the stack and a byte count in the B accumulator. The from-address is typically the start of a token in the command line. The to-address is typically the FCBNAM field of an FCB. The byte count is the total length of the name; the sum of the length of the three tokens (name, . , ext) which comprise it. FMTS expands the wild-card character "*" into a string of "?" of the proper length. FMTS returns a condition byte in the B accumulator as follows:

B=00 unambiguous name
B=01 ambiguous name (wild-cards found)
B=02 bad name (error)

Example of the use of FMTS

```
CMDLIN FCC 'ABC?.*'      length=6 characters

       LDX #FCB+FCBNAM   point to FCB name field
       PSHX
       LDX #CMDLIN       point to command line
       PSHX
       LDA B #6
       FMTS              format name
       INS
       INS               clean stack
       INS
       INS
```

at this point, B=01 and the name field of the FCB contains

ABC?^^^^.??? where "^" indicates a space

53 CMWC

This routine compares strings like CMPC, except that it skips over the wild-card character "?" which matches any character, including a space.

23 OPEND

This routine accesses the directory track on a particular disk and returns a pointer to the first FIB on the disk. It is called with the index register pointing to an FCB which has the drive number set up in FCBDRV and 'DSK' in FCBGDT. The FCBDBA must point to a buffer large enough for one disk sector. The status (FCBSTA) is returned as follows:

```
00=good
01=end of directory found
>1=error condition value
```

If the status is good, the buffer (FCBDBA) contains the first sector of the directory from the indicated disk and FCBIND is initialized to the start of the first FIB. It is up to the user to check that the FIB is not a deleted file. This is done by looking for a space (20 hex) in the first byte of FIBNAM. Hence, if the index register points to an FCB which has FCBGDT, FCBDRV, and FCBDBA properly set, the following code will check for a valid FIB entry.

```
        OPEND               open directory
        TST FCBSTA,X        good status?
        BNE ERROR           no, error!
*
        LDX FCBIND,X        point to FIB
        LDA A 0,X           check first byte
        CMP A #$20          space?
        BEQ NOGOOD          if so, not valid
```

Note that FCBSTA=01 indicates a totally empty disk.

26 GETDR

This routine gets subsequent directory entries from a disk after OPEND has been used. Each call to GETDR will move the pointer FCBIND to the next FIB in the sector buffer. GETDR automatically reads new directory sectors as necessary until the end of the track is encountered. The calling sequence for GETDR is the same as that for OPEND: address of FCB in the index register and status returned in FCBSTA.

27 PUTDR

This routine is used to put a new FIB into a disk directory. It assumes that OPEND and GETDR have been used to find a spot for the new FIB where it will overlay either a deleted FIB or the next unused FIB on the disk. It assumes that the necessary file specification has been placed into the FCB (FCBNAM, FCBTYP, FCBACS, FCBFTS, FCBLTS, and FCBNMS) The index register is pointed to the FCB. PUTDR will copy the FIB entries from the FCB to the disk directory location pointed to by FCBIND. Status information is returned in FCBSTA.

20 OPEN

This routine opens a disk file for an I/O operation. It is called with the index register pointing to an FCB which has been initialized with the appropriate information. To open an existing file, set the following:

```
FCBGDT='DSK'
FCBSTA=0
FCBDTT=0   input
FCBDBA=address of sector buffer
FCBDRV=desired drive
FCBNAM=filename, properly formatted
FCBSCF=00 or FF depending on type of file (space compression)
```

To create a new file, set all of the above plus the following:

```
FCBDTT=FF    output
FCBTYP=desired file type
FCBACS=desired file access code
```

OPEN will check that a new file does not conflict with a file that already exists on the disk and check that a file opened for input actually exists. Error status is returned in FCBSTA. OPEN places the FCB on the active FCB chain (see FCBCHN on base-page). As many open files as desired may be kept in the system, as long as there is a unique FCB for each one.

21 CLOSE

This routine finishes the processing of an active file and removes its FCB from the active file chain. It is called with the address of the FCB in the index register. Error status is returned in FCBSTA. For new files, CLOSE pads the last incomplete sector with nulls (00) so that the file contains all the desired data. CLOSE updates the directory FIB of the file to include the last track/sector used (FIBLTS) and the number of sectors (FIBNMS). Once a file is closed, its FCB space and buffer may be reused.

22 REWD

This routine is actually a CLOSE followed by an OPEN on the same file and using the same FCB. It can only be performed on input files. The effect is to return the file pointers to the start of the file. REWD is called with the index register pointing to the FCB. Error status is returned in FCBSTA.

24 READ

This routine gets a data byte from an file opened for input. Bytes are read sequentially from the file. READ is called with the FCB address in the index register. It returns the data byte in the A accumulator. Error conditions are returned in FCBSTA. If the end of the file is reached, the status will return 08. READ cannot go beyond the end of the file. If space compression is set (FCBSCF=FF), READ will expand the compressed spaces into real spaces. (20 hex)

25 WRITE

This routine places data bytes into a file opened for output. Bytes are written sequentially into the file. WRITE is called with the data byte in the A accumulator and the index register pointing to the FCB. Error conditions are returned in FCBSTA. If space compression is set (FCBSCF=FF (hex)), WRITE will convert spaces (20 hex) into compressed internal format.

Initialization and Warmstart

31 WARMST

This routine returns control to CP/68 from a running program. This is the proper way to terminate a program written to run under the CP/68 system. WARMST will reset the stack pointer to the system stack, close all open files on the FCB chain, clear the free-space entries in base-page, and prompt for a new command.

51 INTDK

This routine does all necessary initialization processes for the disk drives. CP/68 does this on cold-start. The user may use this routine if the drive initialization must be redone from outside CP/68.

Deleting a file

28 DELETE

This routine removes an existing file from a disk. It is called with the index register pointing to an FCB which has FCBGDT='DSK', FCBDRV=desired drive, FCBNAM= filename properly formatted. DELETE checks the access code of the file to be sure that the file may be deleted. If FIBACS>00 , DELETE will issue an error message, set FCBSTA=18, and return. DELETE requires that all open output files on the disk be closed. If there are open output files on the disk, DELETE will issue an error message, set FCBSTA=18 and return. DELETE removes the FIB from the directory by putting a space in the first character of FIBNAM. It links the sectors of the file to the head of the free-space list on that disk. It updates the free-space header link as well. Error conditions are returned in FCBSTA.

Program chaining

29 CHAIN

This routine loads a new program file into memory and starts executing it. It uses LOADB to bring in the new file. CHAIN is called with the index register pointing to an FCB with the desired FCBDRV, FCBNAM, etc. CHAIN moves the data from the user FCB into a system space so that the new file may overlay the user FCB memory. If there was some error, CHAIN will issue an error message and return to the system for a new command. If the file to be CHAINed had no transfer address, this will be flagged as an error. If there was no error, the new file will begin execution at its transfer address.

37 LOADB

This routine loads a binary-format file into memory. The file type (FIBTYP) must be 00 or 01. If it is not, LOADB will issue an error message and return without changing memory. LOADB expects the index register to point to an FCB with FCBGDT='DSK', FCBDRV, and FCBNAM set to the desired file specification. If an error condition is encountered while reading in the new file, LOADB will close the file and return to the system. If the file had a transfer address, it will be stored in the location VALUE in base-page. If there was no transfer address, VALUE will be zero.

User entries

32-36, 38-43 USR1-USR11

These entries in the dispatch table of CP/68 (DSPTAB) are unassigned and are left for the user to add new routines.

FORMAT OF CP/68 BINARY FILES

Binary files under CP/68 (this class includes all transient commands, system utilities, SAVE files, etc.) are stored on disk in a binary format to conserve space. There are two types of data in a binary file: transfer address, and memory data. Each type of data is stored in a block of up to 256 bytes. The format of a transfer address is:

BYTE 1	transfer address mark (16 hex)
BYTE 2-3	transfer address

BYTE 1 memory data mark (02 hex)
BYTE 2-3 memory address
BYTE 4 count of data bytes
BYTE 5--- data bytes exactly as in memory

Memory data is loaded at the address specified with it. There may be more than one transfer address in a file. If so, the last one in the file will be used. The last sector of a binary file will be padded with nulls (00 hex) as necessary to complete the sector. This has no effect on memory loading.

Binary files cannot be transfered to an ASCII device like the PTP or LPT. Similarly, files read from ASCII devices like the PTR or CON are not in the binary format. The system command PIP provides format conversions for these two formats.

he following examples illustrate usage of CP/68 routines to perform seful operations. They are not intended to be optimal programs, but imply to show how easy the CP/68 "extended instructions" make the task f dealing with files, etc.

This example shows how to open, read, write, and close files for nput and output. It is assumed that the user will type filenames in at he console when prompted to do so. Six routines are presented here:

OPENI	open an existing file for input
OPENO	open a new file for output
GETB	get a byte from existing file
OUTB	put a byte out to new file
CLOSI	close the file being read
CLOSO	close the new file

It is assumed that the disk system has been initialized by use of NTDK. Two FCBs are assumed, one for each file in use. In this example, t is assumed that SECSIZ=256 bytes.

```
INFCB   RMB 2          define input FCB
        FCC 'DSK'
        FCB 0
        FCB 0          direction (input)
        FDB INBUF
        RMB 33

OUTFCB  RMB 2          define output FCB
        FCC 'DSK'
        FCB 0
        FCB $FF        direction (output)
        FDB OUTBUF
        RMB 33

INBUF   RMB 256        sector buffer for input
OUTBUF  RMB 256        sector buffer for output
```

he examples assume that the EQUates for FCBs and base-page ɔcations have been set up.

```
OPENI   LDX #INMSG
        PRTMSG         prompt for input filename
        GTCMD          get filename from CONSOLE
        LDX DESCRA
        STX CUCHAR     back up to first token
        LDX #INFCB     point to FCB
*
OPEN2   CLR FCBSTA,X   init. status
        CLR FCBSCF,X   no space compression
```

```
        TST FCBSTA,X  error?
        BNE FILERR    yes, print error message
*
        OPEN          open file
        TST FCBSTA,X  error?
        BNE FILERR    yes
*
        RTS           done!
*
OPENO   LDX #OUTMSG   prompt for file name
        PRTMSG
        GTCMD         get user file name
        LDX DESCRA
        STX CUCHAR    back up to first token
        LDX #OUTFCB   point to FCB
        BRA OPEN2     finish like OPENI
*
FILERR  PRTERR        print error message
        WARMST        return to system
*
*
INMSG   FCC 'INPUT FILE?'
        FCB $04
*
OUTMSG  FCC 'OUTPUT FILE?'
        FCB $04


CLOSI   LDX #INFCB    point to FCB
        CLOSE         close file
        TST FCBSTA,X  error?
        BNE FILERR    yes
*
        RTS
*
CLOSO   LDX #OUTFCB   point to FCB
        CLOSE         close file
        TST FCBSTA,X  error?
        BNE FILERR    yes
*
        RTS


GETB    PSH B         save B accumulator
        PSHX          save index register
        LDX #INFCB    point to FCB
        READ          read a byte from file
*
* the A accumulator now contains the byte read in
*
        LDA B FCBSTA,X check status
```

```
*
        CMP B #8        status=08 is end-file
        BEQ GETB2
*
        BRA FILERR      otherwise, error
*
GETB1   PULX            recover index register
        PUL B           recover B accumulator
        RTS
*
GETB2   set whatever EOF flag is desired
        BRA GETB1


* byte to be written in A accumulator
*
OUTB    PSH B           save B accumulator
        PSHX            save index register
        LDX #OUTFCB     point to FCB
        WRITE           write byte to file
        TST FCBSTA,X    error?
        BNE FILERR      yes
*
        PULX            recover index register
        PUL B           recover B accumulator
        RTS
```

nd for good measure, here is how to rewind the input file.

```
REWIND LDX #INFCB     point to FCB
       REWD           rewind file
       TST FCBSTA,X   error?
       BNE FILERR     yes
*
       RTS
```

ere is another example of the power of CP/68 to do fairly complex tasks n a few simple lines. Suppose the user wishes to have one program load n another whose name is defined in the program. Assume that INFCB and NBUF exist from the previous example.

```
LOADER LDX #FNAME     point to desired file spec.
       STX CUCHAR     store in base page pointer
       LDX #INFCB     point to FCB
       FMTFCB         format file spec. into FCB
       PRTERR         take care of errors
       TST FCBSTA,X   error found?
       BNE QUIT       if so, quit
*
       LOADB          load in new file
```

```
*
* new file must not overlay INFCB or INBUF!!!!
*
       PRTERR        take care of errors
       TST FCBSTA,X  error found?
       BNE QUIT      if so, quit
*
       LDX VALUE     look at transfer address
       BEQ QUIT      if zero, no transfer address
*
       JMP 0,X       go to transfer address
*
QUIT   RTS
*
*
FNAME  FCC '0:MYFILE.BIN'
       FCB $0D       carriage return
```

A somewhat more complex example is this piece of CP/68 which searches a disk directory for an empty FIB location. It assumes an FCB and sector buffer set up like INFCB, etc. The track and sector of the slot (if found) are returned in FCBTRK and FCBSCT. Error status is returned in FCBSTA as follows:

```
00=found slot
01=no space available
>1=error
```

The value TRKSIZ is assumed to be EQUated to the number of sectors in a track of the disk. It is assumed that the A accumulator contains the desired drive to be searched.

```
SEMPTY LDX #INFCB      point to FCB
       STA A FCBDRV,X  save drive number
       TXAB
       LDX #INBUF      get buffer address
       XABX            now X=FCB, A,B=INBUF
       STA A FCBDBA,X  set buffer address into FCB
       STA B FCBDBA+1,X
       CLR FCBSTA,X    init. status
       OPEND           open directory of drive
*
SEMPT2 LDA A FCBSTA,X  check status
       BEQ SEMPT3      status O.K.
*
       CMP A #1        end of directory?
       BEQ SEMPT4      yes
*
       JMP FILERR      otherwise error
*
SEMPT3 LDX FCBIND,X    point to FIB
       LDA A 0,X       check first byte
```

```
        CMP A #$20      space?
        BNE *+3         no
*
        RTS             yes, found an empty FIB
*
        LDX #INFCB      point to FCB
        GETDR           get next FIB from directory
        BRA SEMPT2      keep looking
*
SEMPT4  LDA A FCBSCT,X  get sector number
        CMP A #TRKSIZ   at end of track 0?
        BNE *+3         no, found empty FIB
*
        RTS             yes, no room
*
        CLR FCBSTA,X    return good status
        RTS
```

The next example shows a second way to chain a new program in from another using CP/68. Using the CHAIN SVC, the new program can overlay the one that called it in. The assumptions of an input FCB, etc. are used here.

```
        LDX #MSG        get program name to chain in
        PRTMSG
        GTCMD
        LDX DESCRA      back up to first token
        STX CUCHAR
        LDX #INFCB      point to FCB
        FMTFCB          set name, drive into FCB
        TST FCBSTA,X    error?
        BNE FILERR      if so, quit
*
        CHAIN           bring in new program
*
* CHAIN never returns
* it will either start new program or give
* error message and return to system
*
```

This next example illustrates the active-FCB chain process. It will print on the console the filename of every active FCB in the system.

```
        LDX FCBCHN      get chain header
        BEQ DONE        if=0, no active FCBs
*
LOOP    LDA A #$0D
        STA A FCBNAM+12,X   put c.r. after name
        PSHX            save pointer
        LDX FCBNAM,X    point to name field
        PRTMSG
        PULX            recover FCB pointer
```

```
        LDX FCBNFB,X  get chain pointer
        BNE LOOP      if not=0, loop
*
DONE    RTS
```

This example is the actual code used by the FMTFCB SVC in CP/68. It illustrates the use of NXTOK in parsing a line of text. It also illustrates how register data is passed on the stack to CP/68 SVCs.

```
FMTFCB TSX
        LDX UXH,X     point to FCB
        CLR FCBSTA,X  clear status
        CLR FCBDRV,X  default drive=0
        NXTOK         get a token (assume CUCHAR init.)
        LDA B RC      check RC
        CMP B #3      number?
        BNE PARS2     no
*
        TST VALUE     valid drive no.?
        BNE PARS1     no
*
        LDA A VALUE+1 valid drive no.?
        CMP A #3      (0,1,2,3)
        BHI PARS1     not valid
*
        STA A FCBDRV,X init. drive number
        BRA PARS1A
*
PARS1   TSX
        LDX UXH,X     point back to FCB
        LDA A #21     return error code
        STA A FCBSTA,X
        CLR VALUE
        CLR VALUE+1   return no value
        RTS
*
PARS1A NXTOK          get token from command line
        LDA B RC      check RC
        CMP B #':     colon?
        BNE PARS1     if not, error
*
        NXTOK         get token
        LDA B RC      check RC
PARS2   CMP B #1      unambig. name?
        BEQ PARS4     yes, good
*
PARS3   TSX
        LDX UXH,X     point to FCB
        LDA A #21
        STA A FCBSTA,X  return error code
        RTS
*
```

```
PARS4   LDX DESCRA      point to name
        STX SAVEX       save it in temp. loc.
        LDA A DESCRC    get length of name
        STA A SAVEA     save it in temp. loc.
        NXTOK           get a token
        LDA B RC        check RC
        CMP B #'.       period?
        BNE PARS3       if not, error
*
        INC SAVEA       count the period in length
        NXTOK           get a token
        LDA B RC        check RC
        CMP B #1        unambig. name?
        BNE PARS3       if not, error
*
        LDA B DESCRC    get ext. length
        ADD B SAVEA     get total length
        TSX
        LDX UXH,X       X points to FCB
        LDA A #FCBNAM
        ADDAX           X points to FCBNAM
        PSHX            set up for FMTS
        LDX SAVEX       point to name
        PSHX
        FMTS            format file name
        INS
        INS             clean stack
        INS
        INS
        TST B           error check
        BNE PARS3
*
        RTS
*
SAVEA   RMB 1           temp. locations
SAVEX   RMB 2
```

Part 3

Description of Routines
****

INTRODUCTION

The CP/68 operating system consists of a memory-resident part and transient files which are loaded into memory when needed. The various transient files overlay each other, since only one is ever in use at a given time. The resident part occupies memory from 0100 hex to about 2000 hex. The transients load starting at 2000 hex and occupy no more than 4 K bytes each (up to 3000 hex). A part of base-page is also used (a description of these locations is given elsewhere in this book).

The resident portion of CP/68 consists of five parts:

BIOS- the Basic I/O System
CLI- Command Line Interpreter
DREAD- Directory Read
SFIO- Sequential File I/O
DRIVERS- Disk Drive handlers

There are nine transient commands:

ASSIGN- make device assignments
BOOT- bootstrap system
DELETE- delete a file (part of this command is resident)
INIT- initialize a new disk
LINK- link a system file for BOOT
PIP- Peripheral Interchange Program
SECURITY- manipulate access code of files
SET- manipulate parameters of CON and LPT devices
STATUS- display present device assignments

In addition to these commands, some disk systems require a formatter program.

FORMAT- format a soft-sectored disk

Also included in this book is the Random-Access file package. This transient package of subroutines provides the facilities for random-access file manipulation under CP/68.

BIOS (Basic I/O System)

The BIOS package consists of the software-interrupt handler (SWIHDR) and a set of routines which are called from it. SWIHDR is the only entry point within CP/68; it vectors all requests for system services to their appropriate handler in the system. The system must vector SWI instructions to SWIHDR to enable CP/68 to function. SWIHDR accesses the byte following the SWI instruction to determine the desired system operation. Invalid bytes (CP/68 has 53 valid operations) are vectored to a monitor location trap. Valid bytes are used to index the dispatch table (DSPTAB) to find the 16-bit offset of the system handler. A subroutine jump is made to the handler, passing all the registers on the stack. (SWI placed them there) Upon return, the return address is incremented to skip the operation byte and an RTI instruction returns to the caller.

Extended Instructions in BIOS

BIOS contains a set of system operations which effectively extend the instruction set of the 6800 to include many useful capabilities from the 6809 set. These instructions are described elsewhere in this manual. They are simply listed here.

PSHALL	push all registers
PULALL	pull all registers
TXAB	transfer X to A,B
TABX	transfer A,B to X
XABX	exchange X and A,B
PSHX	push X
PULX	pull X
ADDABX	add A,B to X
ADDXAB	add X to A,B
ADDAX	add A to X
ADDBX	add B to X
SUBABX	subtract A,B from X
SUBXAB	subtract X from A,B
SUBAX	subtract A from X
SUBBX	subtract B from X
INDEX	X=X + A * B
MUL8	A,B = A * B
MUL16	A,B,X = A,B * X
DIV16	A,B = A,B / X (remainder in X)
MOVC	move a character string of given length
CMPC	compare character strings
CMWC	compare character strings with wild-card matches
MOVS	move a character string with 04 hex terminator

FMTS (Format a filename string)

This routine takes a filename string as might be input from the console and formats it into the required CP/68 format. CP/68 wants filenames in the form:

NAME: 8 characters
DOT: period
EXTENSION: 3 characters

FMTS is called with the addresses of the input and output strings on the stack and the length of the input string in the B accumulator. It fills the output string space with blanks (20 hex) and places the dot in the 9th character position. It then moves the name and extension from the input string to the output string. It checks the name and extension for validity as it goes, it also checks for wild-card characters. The B accumulator returns a status code as follows:

00 hex unambiguous, valid name
01 hex ambiguous, valid name
02 hex invalid name

DISPATCH TABLE (DSPTAB)

This table contains the 16-bit signed offsets of each of the CP/68 system routines relative to the SWIHDR handler. Note that $FFFF is -1 in 16-bit binary. The somewhat strange-looking form of the table entries is required since the assembler does not allow unary operators or parentheses in address expressions. For example, *-@PSHAL*$FFFF , could be re-written as -(*-@PSHAL). Note that DSPTAB is also defined as an offset from SWIHDR.

EQTAB (Equipment table) and PDTAB (Physical Device table)

These tables are described in detail elsewhere in this book. They are used by the I/O handler routines, the ASSIGN, and the STATUS transients. Together, they serve to vector I/O requests to the system to the required device handler.

IOHDR (I/O Handler)

This is the central handler for CP/68 I/O requests. It is called with the address of a control block in the index register. IOHDR calls PDSRCH to look through PDTAB for the handler address of the logical device named in the control block. It then calls the handler. Handlers are called with the address of the control block in A,B. If the device name is invalid, IOHDR returns a status of 80 hex which indicates that no such device exists.

PDSRCH

This subroutine is used by IOHDR to access the physical device table. It is called with the address of the control block in the index register. A linear search is performed through PDTAB. If the device name is found, PDSRCH uses the address in PDTAB to point to the EQTAB. There it loads either the input or output handler vector and stores it into the control block. A carry-clear on return indicates that the name was found. A carry-set is returned if the name was not found.

Logical Device Handlers

These routines handle the input and output operations for each of the CP/68 logical devices. Each handler is entered with the address of the control block in A,B. They return that address in the index register.

NULL

The null device simply moves the control block address to the index register and returns.

INLIN (line-oriented input)

This routine handles lines of data from console-type devices. It handles tasks such as fielding "line-delete" and "back-space". It handles echo based on the SET "DX" parameter. It provides the CP/68 input prompt. It also outputs a linefeed for each carriage return.

Calls: INCON, OUTCON

OTLIN (line-oriented output)

This routine handles output of lines to console-type devices. SET parameters such as the null count (NL), line width (WD), paging (DP), ejects (EJ), and pause (PS) are handled in this routine. Detection of a break (any key struck during output) is provided in this routine. This code assumes an ACIA-driven device. The address of the ACIA is derived from the Equipment table.

Calls: INCON, OUTCON

INCON

This routine performs the actual handling of the console ACIA for input. It is called with the index register holding a buffer address. This value is preserved in INCON. The address of the control block is passed on the stack. INCON uses this address to access the EQTAB to get the actual ACIA address. INCON strips the parity bit and returns the character in the A accumulator. INCON will wait for a character.

OUTCON

This routine performs the actual handling of the console ACIA for output. It preserves the index and B registers. It uses the address of the control block from the stack to access EQTAB which gives it the ACIA address. The A accumulator passes the character to be output.

INRDR (line input from papertape reader)

This routine handles input from the papertape reader (PTR) device. It issues the X-ON (11 hex) character to start the reader and uses the X-OFF (13 hex) to turn it off at the end of the line. Nulls (00 hex) are swallowed.

Calls: RDRIN, OUTPCH

OTPCH (line output to papertape punch)

This routine handles line output to the papertape punch (PTP) device. It appends a linefeed (0A hex) and 4 nulls to each line.

Calls: OUTPCH

RDRIN

This subroutine handles the actual input from the ACIA driving the papertape reader. It is identical to INCON except for the stripping of the parity bit.

OUTPCH

This subroutine handles the actual output to the ACIA driving the papertape punch. It is identical to OUTCON.

OTLPT (line output to lineprinter)

This routine outputs a line to the lineprinter device. It assumes a PIA-type interface. The SET parameters for page width (LWD) and page depth (LDP) are handled in this subroutine. OTLPT issues a formfeed (0C hex) to space pages. It automatically adds a linefeed for each line.

Calls: OUTLPT

OUTLPT

This subroutine actually handles output to a PIA port. It preserves the index and B registers. The address of the control block from the stack is used to access EQTAB to get the PIA address. The character to be output is passed in the A accumulator. An acknowledgement signal is expected from the device.

The rest of BIOS is a set of jumps to the other routines forming CP/68. These jumps are necessary for SWIHDR to vector to separately assembled modules. (CLI, Directory read, Sequential File I/O, and Disk Drivers)

The CLI is the heart of CP/68. All command processing passes through it. It contains the routines that load transients and programs, that save memory onto disk, that parse command lines, etc.

Command Table (CMDTAB)

This table contains all the commands directly recognized by CP/68. Each table entry consists of the first three characters of the command name and the address of the command handler. Hence, all CP/68 command names can be abbreviated to their first three characters. A zero marks the end of the table.

Character Table (CHRTAB)

This table is used by the parsing routines (NXTOK) to evaluate a character for the type of token it could be in. Characters from the space (20 hex) to underline (5F hex) in the ASCII set have an entry in the table. Each entry is a byte where each bit has a significance as follows:

```
Bit 7   Alphabetic
Bit 6   Decimal digit
Bit 5   unused
Bit 4   unused
Bit 3   delimiter
Bit 2   Hexadecimal digit
Bit 1   Wild-card character
```

A set bit indicates that the character is a member of the class. For example, the letter "A" has the entry 82 hex. This means that it is both an alphabetic character and a hex digit. Note that the wild-card characters are declared alphabetic (81 hex).

CLI Main loop

There are two entries to CLI, called COLDST and WARMST. There is a jump to COLDST at the beginning of BIOS (start of CP/68). This is the starting location of the system. WARMST is the return to the system, and it is reached through SWIHDR. COLDST performs the initialization steps for the system. The stack pointer is set to the internal stack space. The SUBMIT flag is cleared (no SUBMIT in process). The console and TTY ACIAs are initialized. The set-up for the CON device is:

```
Counter divide-  16
Word select-     8 bits + 1 stop, no parity
```

Interrupts- disabled

The set-up for the TTY device is:

Counter divide- 16
Word select- 7 bits + 2 stop, even parity
Interrupts- disabled

The lineprinter PIA is initialized as follows:

A side: undefined
B side: output, CB1 active low input, IRQ disabled
CB2 output

INITDK is called to initialize the disk hardware. The console control block CONRCB is initialized and the start-up banner is printed. The header of the active-file chain is initialized. Processing now begins the usual CLI loop.

WARMST also sets the stack pointer and clears the SUBMIT flag. It then looks through the active-file chain, closing all files that it finds. It then enters the usual CLI loop.

WARM3 marks the start of the command-processing loop. First, the four free-space headers are cleared. Now a command line is input using GTCMD. This line might come from the console or from a SUBMIT file. GTCMD automatically parses the first token from the line. If it is an ambiguous name (wild-cards), it is a format error. If it is a number, it is assumed to be the drive number of a filename. Otherwise, it is an unambiguous name which might be a command or else a filename on drive 0.

The command table is searched to determine if the name is a command. If the name is found, control jumps the the processor for that command which returns to WARM3 when it completes. If the name is not found, or if this is a filename not on drive 0, the system routine (LODCMD) brings the named file into memory. Since LODCMD does its own parsing of file names, the pointers are first returned to the start of the command line. If a transfer address was loaded, control jumps to that address. If no transfer address was found, or after the loaded process returns, control returns to WARM3 for a new command.

Calls: LODCMD

PRTMSG

This routine prints a message on the console and is used by all the CP/68 routines for printing error messages and prompts. It is called with the address of the text string in the index register. The string must be terminated with either a carriage return (0D hex) or a string terminator (04 hex). The carriage-return causes an automatic linefeed, the string

terminator does not.

PRTERR

This routine prints a formatted error message on the console. It is called with the address of a control block in the index register. It tests the status byte in the control block for error conditions. If there was no error, it prints nothing. If the status byte is nonzero, it converts the byte to hex and stores it in the error message field DERNUM. The device name is taken from the control block and stored in DEVNAM. Finally, the error message is printed.

GTCMD

GTCMD is called to input a line of text from the user. Based on the SUBMIT flag SUBFLG, the line might come from the console or from an open SUBMIT file. If SUBFLG is cleared, GTCMD reads a line from the console. If SUBFLG is set, GTCMD reads a line from the open SUBMIT file, using the file-control block SUBFCB. If reading from a file, the special characters "&" and 04 hex (control-D) are processed. The control-D indicates the end of the SUBMIT file; the file is closed, SUBFLG is cleared, and a line is input from the console. The "&" indicates diversion in a SUBMIT file, one line is taken from the console without upsetting the file or SUBFLG. No matter where the line came from, GTCMD always goes into the parsing routine NXTOK to find the first token on the line.

Calls: NXTOK

Flags: SUBFLG

NXTOK (parsing tokens)

This routine performs the parsing function on a CP/68 command line. Each time it is called it determines the next lexical token of the command line. There are six types of tokens which are recognized:

Multi-character strings-	Unambiguous name
	Ambiguous name
	Number
Single characters-	Delimiter
	Carriage return
	Error (undefined)

NXTOK uses the pointer CUCHAR to point to the starting point on the line to begin parsing. NXTOK moves CUCHAR to point just beyond the end of

the present token. NXTOK returns four values for each token. DESCRA is a pointer to the first character in the token. DESCRC is a count of the length of the token. RC is a code for the type of token. CLASS is a code for major classification of the token.

NXTOK first skips over any blanks up to the first non-blank character. If the character is less than 20 hex, it is either a carriage return or undefined. If it is greater than 5F hex, it is undefined. This means that lower-case characters are not recognized. Next, NXTOK calls GCHRTB which looks up the character in CHRTAB. If the character is alphabetic, NSCAN is called to parse the name. If the character is a decimal digit, DSCAN is called to parse the decimal number. If the character is neither, and it is not a delimiter, it is an error. If it is a delimiter, NXTOK checks for a "$" character. If found, HSCAN is called to parse a hexadecimal number. Otherwise, the delimiter token is returned.

Calls: GCHRTB, NSCAN, DSCAN, HSCAN

DSCAN

This routine parses a decimal string. It looks at characters from the command line one at a time until a non-decimal digit is found. The pointers are decremented to the last decimal digit and it is checked for length (since CP/68 works with 16-bit numbers, it can accept nothing larger than 65535). CVDB is called to convert the decimal string into binary which is returned in VALUE.

Called by: NXTOK

Calls: GCHRTB, CVDB

NSCAN

This routine parses an alphanumeric string. It looks at characters from the command line one at a time until a non-alphanumeric character is found. The pointers are then decremented to point to the last alphanumeric character in the string. The B accumulator is used to indicate if a wild-card character was found in the name string.

Called by: NXTOK

Calls: GCHRTB

HSCAN

This routine parses a hexadecimal number as indicated by a leading dollar sign ($). It looks at characters from the command line one at a time until a non-hexadecimal digit is found. The pointers are then decremented to point to the last hexadecimal digit in the string and the length is checked (since CP/68 can accept numbers up to $FFFF). CVHB is called to convert the hex string into binary which is returned in VALUE.

Called by: NXTOK

Calls: GCHRTB, CVHB

GCHRTB

This routine accepts a character in the A accumulator and uses it to index the character table CHRTAB. The entry from the table is returned in the A accumulator.

Called by: NXTOK, NSCAN, DSCAN, HSCAN

Tables: CHRTAB

CVHB

This routine converts a hexadecimal string into binary. On entry, DESCRA points to the start of the string and DESCRC is the number of characters in the string. It returns the 16-bit unsigned binary value in the index register.

Called by: HSCAN

CVDB

This routine converts a decimal string into binary. Its calling sequence is identical to CVHB.

Called by: DSCAN

All command processing routines are called as subroutines from the CLI loop.

JMPCMD

This routine processes the JUMP command. It uses NXTOK to parse the jump address. It removes the return address (JMPCMD was called as a subroutine) from the stack and executes a jump to the address specified in the command line. If the routine jumped to executes an RTS, it will return to the CLI loop. A "safer" return would be to issue a WARMST call.

Transient Command Processor

The set of CP/68 commands processed by transients:

ASSIGN, BOOT, DELETE, LINK, PIP, SECURITY, SET, STATUS

must load the required file into the transient space. This is accomplished by using a "dummy command" which effectively forces the filename of the transient command to become the command line. LODCMD is called to bring the transient into memory. For the transients that require it, the address of PDTAB is passed in the A and B accumulators.

Calls: LODCMD

SUBCMD (SUBMIT command processor)

This routine processes the SUBMIT command. It uses FMTFCB to parse a filename from the command line into the SUBMIT FCB (SUBFCB). Blank expansion is turned on and the file is opened. The filetype is checked to insure that the file is a text file. The SUBMIT flag is set, indicating to GTCMD that lines should now come from the file, not the console.

SAVCMD (SAVE command processor)

This routine processes the SAVE command. It first initializes the control block SAVFCB as a type 0 file. FMTFCB is used to parse the filename into SAVFCB. The starting address is then parsed and saved in SAVEX. The ending address is parsed and saved in SAVEX1. If this is the end of line, then no transfer address is desired. If there is a delimiter, then a transfer address is parsed, the filetype is made 1, and a transfer-address block is written to the file. Next, data records consisting of 256 data bytes each are written out to the file. When the ending address is reached, the last data block is written out and the file is closed.

LODCMD (LOAD command processor)

This routine loads a file into memory. It processes the LOAD command and is used by the CLI loop and the transient command processor as well. It uses FMTFCB to parse the filename and then uses LOADB to actually load the file into memory.

Called by: CLI loop, Transient processing, INICMD

Calls: LOADB

LOADB

This routine actually loads a memory-image file (produced by SAVE) into memory. The file must be type 0 or 1 (memory-image). The load process opens the file and looks for either data blocks or transfer address blocks. Data blocks contain their load address, so the following data is stored into the indicated address. Transfer address blocks store their address into VALUE. Hence, the last transfer address found in the file will be used.

Called by: LODCMD, CHAIN

RENCMD (RENAME Command processor)

This routine processes the RENAME command. FMTFCB is first used to put the old filename into SAVFCB. SFILE is called to search the directory for this file. If found, the access code is checked to see whether this file is rename-able. If so, the second filename (the new one) is parsed. Note that the second filename can have no drive number, since the first drive number is assumed. Pointers to the directory entry of the old file are stored in SAVFCB. SFILE is called with the new filename to insure that it does not duplicate an existing name. If there is no duplication, the

directory entry for the old filename is re-accessed and the new name field is written into it.

Calls: SFILE

INICMD (INITIALIZE Command Processor)

This routine processes the INITIALIZE command. It parses the drive number and checks it for validity. LODCMD is used to bring the transient code for INIT. into memory. The drive number is passed in the A accumulator and control is given to the transient code. When it is complete, it returns to the CLI loop.

Calls: LODCMD

DIRCMD (DIRECTORY Command Processor)

This routine processes the DIRECTORY command. It begins by formatting ALLFIL into a temporary BUFLIN. ALLFIL is a wild-card specification which matches all filenames. The lineprinter flag LPTFLG is cleared to direct output to the console. A check is made for the lineprinter switch /L . If found, the lineprinter flag LPTFLG is set. Otherwise, DIRCMD looks for a drive number. If a number is found, it is checked for validity and if it is valid it is stored in SAVEA. Next, DIRCMD looks for a file specification. This file specification may contain wild-cards. If a file specification is found, it is formatted into BUFLIN. The number of sectors used (NSEC) is cleared. If LPTFLG is set, the output is re-directed to the LPT device. The drive number is recovered from SAVEA and converted to ASCII. The header messages are printed. The directory of the desired drive is opened.

DIRCMD now loops through each directory block on the given disk. It compares each file on the disk with the name in BUFLIN. If they do not match (including wild-cards), DIRCMD looks at the next file in the directory. If a match is found, the data from the directory block is formatted into a string for output. The string is printed and DIRCMD looks at the next file. When the end of the directory is found, the number of sectors used (the sum of the number of sectors of each file which matched) is converted to ASCII and the finishing message is printed.

Imbedded in DIRCMD is a routine called CVBTD. This routine converts a 16-bit unsigned binary number to ASCII. The number is passed in the A and B accumulators. The address of the place to form the ASCII text is passed in the index register. CVBTD generates five characters.

CHAIN

This routine provides CP/68 the facility to load and run a transient file from an executing program. It works by moving the necessary information from the user's FCB to the system SAVFCB. The user's FCB address is passed in the index register. By moving to SAVFCB, the new program can overlay the user's FCB. CHAIN calls LODCMD to bring the new file into memory. If a transfer address is found in the new file, control jumps to it. Otherwise, control returns to the CLI loop.

Calls: LODCMD

SEMPTY

This routine is used to search a disk directory for an empty slot. It looks through the directory for either a directory block with a blank as its first character (indicates a deleted file) or the end of the directory. If a usable directory block is found, SEMPTY returns a status of 0. If no usable block is found, a status of 1 is returned. SEMPTY uses a system control block SYSFCB. It is called with the drive to search in the A accumulator. It returns the pointers to the directory block in SYSFCB. (FCBTRK, FCBSCT, and FCBIND) The status is returned in FCBSTA.

Called by: OPEN (sequential file I/O)

SFILE

This routine searches a disk directory for a given, non-ambiguous file. It is called with the address of a control block in the index register. This FCB contains the drive and filename of the file to be searched. SFILE returns status in the supplied FCB. A status of 0 indicates the file was found. A status of 1 indicates the file was not found. FCBIND in the supplied FCB points to the directory block. SFILE uses SYSFCB to manipulate the directory.

Called by: OPEN, CLOSE (sequential file I/O), RENCMD, DELETE

DELETE (Resident part of DELETE command)

This routine handles the removal of a file from a disk. It is called with the address of an FCB in the index register. This FCB contains the filename and drive of the file to be deleted. First, SFILE is called to locate the file in the directory. The access code is checked to see if this file may be deleted. If so, all the active FCBs are checked to see if there are any open files on this disk. If there are, no file deletes may be performed on the disk, since this might corrupt the linkages of the sectors. If there are no active files on this disk, the directory entry of the file is read in. The first and last track/sector pointers are saved. A blank is inserted into the name field in the directory. The present header of the free-space list on this disk is saved. The first track/sector of the file becomes the head of the free-space list. The last track/sector of the file is linked to the old free-space header. This puts the sectors from the deleted file back onto the free-space list. The free-space sector is updated to match this.

Calls: SFILE

FMTFCB

This routine parses a file specification from the command line and places the result into a supplied FCB. The address of the FCB is passed in the index register. The pointer CUCHAR indicates the beginning of the file specification. FMTFCB first looks for a drive number. If none is found, drive 0 is assumed. If a number is found, it is checked for validity. FMTFCB expects an unambiguous name. (no wild-cards) If a syntax error is found while parsing, 21 hex is returned in the FCBSTA field of the FCB.

DIRECTORY-READ Routines

This set of routines provides the means to read and change a disk directory under CP/68. It consists of three entries: OPEND, GETDR, and PUTDR. A CP/68 directory is a sequence of 32-byte directory blocks stored on the first track of the disk. The end of the directory is marked by a directory block whose first character is a zero. If the first character is a blank (20 hex), this directory block is assumed to have been deleted and new files will over-write it.

OPEND

This entry opens a disk directory for use. It positions the drive to the first track (directory) and reads in the first sector of the directory. The first character of the directory sector is tested. If it is zero, the disk directory is empty and a status of 01 hex is returned, indicating

that the end of the directory was found. If it is not zero, a zero status is returned. OPEND is called with the address of a user FCB in the index register. The FCB must have the drive number set and the device-type must be set to DSK. It returns status information in the FCB.

GETDR

This entry reads directory blocks from an open directory. OPEND must be called prior to calling GETDR. GETDR moves the pointers to the directory 32 bytes forward each time it is called. This effectively accesses the directory block for the next file on the disk. GETDR will read a new sector when it finishes the previous one. It will return a status of 00 hex if it finds a good file block in the directory. It will return a status of 01 hex if it finds the end of the directory. Its calling sequence is the same as that of OPEND.

PUTDR

This entry updates a directory block that has been found with OPEND and GETDR. The changes to the file directory data are made to the copy in the sector buffer used with OPEND and GETDR. Calling PUTDR with the address of the FCB in the index register will re-write the directory sector into the directory, making the desired updates.

SEQUENTIAL-FILE I/O Routines

These routines handle sequential files under CP/68. They direct the directory-routines and the drivers to form a file-management system. There are five routines: OPEN, CLOSE, READ, WRITE, and REWD. Each is called with the index register pointing to an FCB. Those routines which pass characters (READ, WRITE) use the A accumulator. These routines also handle space-compression for text files.

OPEN

OPEN prepares files for use under CP/68. It first checks that the file is not already open, then it determines whether the file is to be opened for input or output. The in/out decision is based on the FCBDTT byte in the FCB.

Input files are checked against the disk directory to see if the file already exists. The system subroutine SFILE performs this check. Next, OPEN moves the file pointers, type, etc. from the directory to the FCB. The first sector of the file is read in; the forward and backward sector links are put into the FCB. Finally the FCB is added to the

active-FCB chain.

Output files are processed differently. SFILE is called to check that the new filename does not duplicate an already existing file. Next, the system subroutine SEMPTY is called to find an available directory block for the new file. The FCBNMS (number of sectors), FCBLTS (last track/sector), and FCBBAK (back pointers) fields in the FCB are cleared. The free-space header for the desired disk is accessed. If it is nonzero, this is the track/sector of the next available sector. If it is zero, the free-space sector (link sector) is read and the header is updated. The free-sector is checked to see that it is not the end of the disk (0,0). The FCBFTS (first track/sector) field in the FCB is initialized to the free sector and the directory entry is written using PUTDR. The free sector is read in and the free-space header is updated to be the next available sector. Finally, the FCB is added to the active-FCB chain.

Calls: SFILE, SEMPTY

CLOSE

This routine finishes the processing of a file. First CLOSE checks that the FCB is open. If it is found in the active-FCB chain, it is removed from the chain. If it was an input file, CLOSE is finished. For output files, CLOSE must write out the last sector. It uses SFILE to find the directory entry for the file and updates the FCBLTS (last track/sector) and FCBNMS (number of sectors) entries. The free-space record is updated. This completes the CLOSE process.

Calls: SFILE

READ

This routine gets a byte from an open input file. It checks to see if the desired byte is in the sector buffer already. If it isn't, a new sector is read in and the forward and backward links are updated; the byte is accessed from the buffer. If no space-compression is required, the file pointer (FCBIND) is incremented and the data byte is returned. If space-compression is required, a test is made of the data byte. If the byte is positive (high-order bit is zero), the data byte is returned unchanged. If the byte is negative (high-order bit set), the byte is a compressed space. The data byte is actually the negative count of the number of spaces desired. The data byte is incremented and restored to the buffer while a space (20 hex) is returned. When the data byte reaches 00 hex, the last space is returned and the file pointers are moved. Until then, spaces are returned while the file pointer stays in the same point in the sector buffer.

WRITE

This routine writes data bytes to an open disk file. It first checks that the file is open for output; next it checks to see if the end of the sector buffer has been reached. If it has, the present sector buffer is written to the disk. The number of sectors in the file (FCBNMS) is incremented; the free-space header is updated, as are the forward and backward file pointers (FCBFWD and FCBBAK). A new sector is read in from the free-space chain and linked to the file. In either case, the next step is to store the data byte into the sector buffer. If no space-compression is being done, WRITE is completed. If space-compression is being done, and if the data byte is a space (20 hex), the present value of the data byte in the file is checked. If it is negative (compressed space), the value is decremented (one more space) and restored. If it is not negative, a single compressed space (FF hex) is stored into the file. This completes WRITE.

REWD

This routine rewinds an input file to its starting point. Effectively, REWD is a CLOSE followed by an OPEN.

DRIVER Routines

These routines provide the interface between CP/68 and the disk hardware. Three entries are needed: INITDK, RDSEC, and WTSEC. The exact mechanism of these routines depends on the hardware being used.

INITDK

This routine performs all necessary initialization required by the disk system. This may include initializing peripheral interfaces, setting memory flags, calling ROM routines, etc. It is called with no parameters.

RDSEC

This routine reads a desired sector from the disk. It is called with the address of an FCB in the A and B accumulators. The FCB contains the drive, track, and sector pointers. It also contains a pointer to the buffer area. The status of the read must be returned in the FCB. It should also be returned in the A accumulator. Since these routines are called from software interrupts, they must change the stacked-value of the accumulator in order to return it. RDSEC must detect disk errors and return appropriate error status numbers.

WTSEC

This routine writes a desired sector to the disk. It is called with the address of an FCB in the A and B accumulators. The FCB holds the drive, track, sector, and sector-buffer pointers. The status should be returned in the same manner as RDSEC.

Transient Commands
****

ASSIGN Transient Command

This routine processes the ASSIGN command from CP/68. It re-directs a logical device by modifying the physical-device table entry (PDTAB) of a given device. PDTAB entries consist of 7 bytes. The first three bytes are the name of the device. The next two bytes are a pointer to the appropriate entry in the equipment table (EQTAB) where the device handler addresses are found. The last two bytes are also a pointer to the EQTAB. ASSIGN modifies the first pointer field, but the second pointer is left intact so that other routines (such as STATUS) can find the original device assignment.

When ASSIGN is called from the command-interpreter, the address of PDTAB is passed in the A and B accumulators. ASSIGN then proceeds to parse the command line, obtaining the names of the devices to be assigned. The device to be assigned is stored in DEV1, the device to which it is being assigned is stored in DEV2. The subroutine PDSRCH is used to check the names in DEV1 and DEV2 against the names in PDTAB to insure that both are valid device names.

If DEV1=DEV2, the second pointer field of the name is copied into the first pointer field of the name. If DEV1 is different from DEV2, then both names are checked with PDSRCH, and the second pointer field of DEV2 is copied into the first pointer field of DEV1. Note that even though DEV2 may have been re-assigned itself, the second pointer field retains the initial value.

Called by: CLI
Calls: PDSRCH
Tables: PDTAB

PDSRCH

ASSIGN uses this routine to check device names for validity. It searches the physical-device table (PDTAB) for a device name whose address is passed in the index register. The end of PDTAB is marked with a zero. PDSRCH returns with carry-set if the device was not found, and with

carry-clear if the name was found.

Called by: ASSIGN
Calls: none
Tables: PDTAB

BOOT Transient Command

This routine bootstraps a system file from drive 0 using no system support. It assumes that the disk in drive 0 has had a bootable file linked on it (See LINK). It is written to be ROMable, with all necessary RAM locations in COMMON storage. It also uses its own stack space.

The first step BOOT performs is to initialize the disk drives. This process varies depending upon the hardware requirements. The next step is to read in the link sector. (track 0 sector 1) The last six bytes of this sector contain special information.

SECSIZ-6 First track of linked file
-5 First sector of linked file
-4 Last track of linked file
-3 Last sector of linked file

SECSIZ-2,1 Free-space pointer

The track/sector pointers define the linked file.

BOOT loads the desired file into memory just like the system LOADB routine does. The marker 16 hex indicates a transfer-address block, the marker 02 hex indicates a data block. The loading process continues until the last sector of the file (as determined from the link sector) has been loaded. The program then jumps to the transfer address read from the booted file. Finding a null (00 hex) while searching for a data block will also indicate the end of the file and will cause a transfer to the start address read from the file.

Called by: CLI
Calls: GETBYT, RDSEC

GETBYT subroutine

This routine is used by BOOT to read in the desired file. It returns data bytes in the A accumulator. When necessary it calls RDSEC to get a new data sector from disk. When GETBYT finishes the last data byte of the last sector of the file, it jumps to the spot in BOOT which indicates an end-of-file condition.

Called by: BOOT
Calls: RDSEC

RDSEC Routine

RDSEC is the routine used to read individual sectors from the disk. It is called with the desired track in accumulator B, the desired sector in accumulator A, and the address of a buffer in the index register. RDSEC assumes drive 0. The actual mechanism of RDSEC depends on the hardware used to control the disks.

Called by: BOOT, GETBYT

DELETE Transient Command

This transient routine is used in conjunction with the resident DELETE code to handle the removal of files from the disk. The resident code actual performs the disk update, this transient handles set-up for it and also takes care of wild-card names, check-prompting, and other tasks.

DELETE first accepts a filename and tries to format the name into its internal SYSFCB. Since there may be wild-cards in the name, a temporary buffer called TEMP is used to hold the name. If the name parses as a good filename, the next step is to search the desired disk directory for a file whose name matches the given name in TEMP. If such a file is found, DELETE forms a prompt line with the file name and waits for a user response. If the response is "Y", the file is set up for the resident DELETE and is then erased from the disk. After the file is erased, or if the response was not "Y", the transient continues to search the disk directory for further matches. If more are found, they will each be prompted in turn. When the end of the directory is found, DELETE will prompt for a new filename. Entering an ESCAPE character returns the system to the command level.

Called by: CLI
Calls: none

INITIALIZE Transient Command

This routine builds the necessary data structure for CP/68 on a blank disk. Soft-sectored disks must have been previously formatted before using this routine on them.

INITIALIZE first prompts the user that it is ready to initialize a disk in a given drive. The drive number is passed in the A accumulator from the CLI. If the user responds "Y", the initialization process begins. If the response is not "Y", the program returns to the CLI.

Initialization begins by writing the link sector. The last two bytes of this sector are set to point to track 1, sector 1 (the start of the

free-space). The remainder of track 0 (directory) is cleared. The rest of the sectors on disk (tracks 1 and above) are linked together into a free-space chain. The first two bytes of each sector point to the next sector. The third and fourth bytes point back to the previous sector. The remainder of the sector is cleared. The forward pointer of the last sector on the disk points to 0,0. The sectors need not be contiguous. A table called TBL is used to initialize the disk to an interleave pattern determined to provide the fastest access times for files. This table is entered with a logical sector number, it returns the physical sector number on the given track. The subroutine GETSC performs the lookup in TBL. The subroutine WRTBLK is used to write data sectors onto the disk. If a disk error occurs, the initialization process is aborted with an error message that indicates the sector and track of the bad spot on the disk.

Called by: CLI
Calls: GETSC, WRTBLK

GETSC

This subroutine converts a logical sector number into a physical sector number, using an interleave table TBL.

Called by: INITIALIZE
Tables: TBL

WRTBLK

This subroutine writes a data sector onto the disk. An internal control block FCBSPC is used to direct the writing. Errors are trapped to WRTERR which outputs the track, sector, and error numbers in hex.

Called by: INITIALIZE

LINK Transient Command

This routine is used to set the pointers in the link sector to point to a desired file. This is typically a CP/68 system file, but it can be any binary file which is to be bootstrapable.

The first step is to prompt the user for a file name. The name is parsed to be sure that it is a valid, non-ambiguous file name. LINK then looks up the file name in the disk directory. If found, the first and last tracks and sectors are recovered from the directory and placed in the internal SYSFCB. If the file is not found, or if it was not a valid filename, LINK gives an error message and returns to the CLI. If found, the link sector of the disk is read, the pointers updated to those from

the directory, and the link sector is re-written to the disk. It then returns to the CLI

Called by: CLI

PIP Transient Command

This routine handles all forms of data manipulation from one device or file to any other device or file. PIP (Peripheral Interchange Program) handles such diverse tasks as file concatenation, disk copy, binary-to-MIKBUG conversion, etc. It has several sections which perform different operations.

DEVTAB

This table lists the various devices supported by CP/68 and has the addresses of handlers for them. This differs from PDTAB and EQTAB in that PIP uses character-by-character I/O, not line-oriented I/O as used in the rest of CP/68. Each entry in DEVTAB consists of 11 bytes. The first three bytes are the device name. The rest of the entry is a set of four addresses, each two bytes. The first address is a handler for device "open". The second address is a handler for device "close". The third address is a handler for device character read. The last address is a handler for device character write. If one of these addresses is zero, it indicates that the device cannot perform the desired operation. (Read from line printer, etc.) The end of the table is marked with a zero.

CHARACTER-ORIENTED DEVICE HANDLERS

These short subroutines handle the various devices under CP/68 so that they can provide character-by-character I/O. The "open" routines check that the device is capable of the desired operation. The "open" for the lineprinter automatically emits a form-feed (0C hex). The "close" routines for devices like the paper-tape punch automatically add control-D (04 hex) to indicate end-file. The "read" routines for devices like the paper-tape reader and teletype check for control-D and return end-file status when it is found. All the routines are called with the address of a control block (one of the internal FCBs) in INHND for input and OUTHND for output.

DLKUP

This subroutine performs the lookup of a device name in DEVTAB. The address of the device name is passed in the index register. Carry-set on return indicates that the name was not found. Carry-clear indicates that the name was found and the address of the table entry is in the index

register.

PIP itself

The main body of PIP parses the command lines and determines the necessary processing. The first step is initialization of the input and output FCBs. The device is assumed to be disk 0 unless otherwise specified. A blank is placed in the first character of the filename field. PIP next processes the left side of the command line. If a number is found, it is checked for validity as a drive. If an error is found, PIP reprompts for another command. Otherwise, the program tries to complete the file name parsing. A valid filename is parsed into the input OUTFCB. If no number was found, the entry might be a file on drive 0 or a device name. DLKUP is used to check whether the entry is a device name. If not, the entry is formatted as a file name; if it is, the device name is placed in OUTFCB. The address of the device handler is placed in OUTHND.

PIP next looks for a slash (/) that indicates the presence of switches. If a slash is found, the switches are checked and appropriate flags set. Switches are separated with slashes. Parsing of the output portion of the command line ends with the equals sign in the line.

The output portion of the command line could also be a drive specification only (number followed by a colon). If this is the case, a flag is set to indicate that a form of disk-copy is requested (PIPFLG).

The input portion of the command line (right of the equals sign) is parsed much the same as the output side, except that no switches are allowed. Ambiguous filenames (with wild-cards) are allowed if in a file-copy (PIPFLG set).

Once the command line has been parsed, the transfer of data can begin. The character-oriented device handlers are used to move data from the input device to the output device. Upon completion of the transfer, PIP checks the command line for a comma or other delimiter on the right. If found, this indicates another input source is to be concatenated. The source specifier is parsed and if valid, its data is also transfered.

I/O errors during transfer are indicated, but the processing continues. Note also that since transfers are buffered by the handlers, there will be a one line lag between input and output.

Upon completion of data transfer, PIP reprompts for a new command line after issuing a "DONE" message. An ESCAPE character will allow return to command level.

DTDCPY

This routine is called when PIP determines that the form

    drive: = drive:

has been commanded. This routine performs a direct sector-for-sector copy from one disk to another. A prompt is issued which indicates the direction of copy and gives the user a chance to correct mistakes in the command.

FILCPY

This routine is called when PIP determines that the form

drive:= drive: wildcard name

has been commanded. The wild-card filename is moved into temporary storage TMPBUF. The disk directory is searched for filenames which match the name. If a match is found, the name is echoed and the user is prompted for a response. If the response is "Y", the file is copied. After the copy, or if the response was not "Y", further matches are sought in the directory. Each match is prompted in turn until the directory is exhausted.

HEXFRM

This routine converts the internal binary-format of program files into MIKBUG or hexadecimal format. It is called when the H switch (HFLAG) is set by PIP.

BINFRM

This routine converts MIKBUG or hexadecimal-format data into the internal CP/68 binary format. It is called when the B switch (BFLAG) is set by PIP.

SECURITY Transient Command

This routine is used to change the access code of a given file. It first parses the filename passed to it by the CLI. This name is looked up in the disk directory. If not found, an error message is returned and the CLI is resumed. If the file is found, its directory information is retained in the internal SYSFCB. The command line is parsed for a comma followed by a number. If found, and if the number is less than 256, the number is placed into the directory access entry of the named file and the directory is updated. If an error was found, the program simply returns to CLI without changing the directory.

Called by: CLI

SET Transient Command

This routine processes the SET command. It manipulates the CONsole and LPT parameters in base-page. The set of legal parameter names is contained in the table SETAB. Each entry consists of 4 bytes. The first two bytes are the 2-character name of the parameter. The second two bytes are the address of this parameter. Two bytes are used because not all versions of CP/68 place the parameters in base-page. The subroutine SETSRC searches this table for the parameter whose name is contained in the index register. Carry-clear indicates that the parameter was found in the table and that its address is in the index register. Carry-set indicates that the name was not found.

The normal case of SET is PAR=number. In this case, the value of "number" is stored at the address recovered from SETAB based on "PAR". There are two special cases in SET. If PAR=DX, the appropriate values are not numbers but "F" or "H" (full or half-duplex). SET checks for these responses and stores FF hex into the DX parameter address for half-duplex and 00 hex for full-duplex. If PAR=PS, the appropriate values are "Y" or "N" (pause Yes or No). SET checks for these responses and stores FF hex into the PS parameter address for pause-off and 00 hex for pause-on.

Called by: CLI

STATUS Transient Command

This routine prints out the present state of logical/physical device assignments. It is called with the address of the physical device table (PDTAB) in the A and B accumulators. It works by taking the device name of an entry in the table and looking at its two address pointers. If they are the same, the device has not been re-assigned and so it can be printed as

DEV = DEV

If the pointers differ, it indicates that a re-assignment has been

done. PDTAB is searched for an entry whose second address pointer matches the first address pointer of our given entry. When found, its device name is the one to which the given device has been re-assigned. Therefore, if DEV1 is the given device name, and DEV2 is the name of the entry whose second address matched DEV1's first address pointer, STATUS prints

DEV1 = DEV2

STATUS performs this operation for all devices in PDTAB and then returns to CLI.

Called by: CLI

FORMAT Transient Utility

Those versions of CP/68 which utilize soft-sectored disks require a program which writes the necessary format data onto new diskettes. This information must be on the disk prior to initialization. It usually needs to be written only once.

The FORMAT program consists of three parts: the driver, the track-build subroutine, and the track-write subroutine. The driver and track-build sections are the same for all hardware (on 5-inch disks using 128-byte sectors). The track-write section varies for different hardware configurations.

DRIVER

This routine gets a drive number from the user. It checks this number for validity and issues another prompt to the user. The second prompt allows the user to change disks or to abort the formatting process. The rest of the driver is a loop which calls TRKBLD and then TRKWRT for each track on the disk.

TRKBLD

This routine builds an image of an entire formatted track in memory (TRKBUF). TRKBLD assumes 128-byte sectors, 18 sectors per track, and a Western Digital 1771 disk controller. The track format is:

```
GAP      8 bytes  of FF hex

GAP      7 bytes  of FF hex       sector starts here
SYNC.    4 bytes  of 00 hex
ID-MARK  1 byte   of FE hex
TRACK #  1 byte (track number)
         1 byte   of 00 hex
SECTOR   1 byte (sector number)
         1 byte   of 00 hex
LENGTH   1 byte   of 00 hex (128 bytes)
CRC      1 byte   of F7 hex
GAP      11 bytes of FF hex
SYNC     6  bytes of 00 hex
D-ADDR   1 byte   of FB hex
DATA     128 bytes (00 hex)
CRC      1 bytes  of F7 hex
PAD      1 byte   of FF hex       end of sector

(repeat for 18 sectors)

GAP     400 bytes of FF hex
```

Track numbers are set by the driver in a location called TRACK. Sector numbers are set in a location called SECTOR. TRKBLD needs at least 3400 bytes for its track image.

TRKWRT

This subroutine is called by the driver to transfer the track image built by TRKBLD to the disk. TRKWRT must position the desired drive to the desired track. The drive number is found in the CP/68 location VALUE. The track number is found in TRACK. After positioning the drive, TRKWRT must do a track-write operation. The exact mechanism of this operation depends upon the hardware in use.

Part 5

Random-access files

This section discusses the random-file support package provided with the CP/68 operating system. You can link it to STRUBAL+ or assembly programs which run under CP/68 and which will manipulate random-access files.

WHAT ARE RANDOM-ACCESS FILES?

Random-access files are a special type of file structure. There are two major differences between the normal CP/68 sequential file and the random-access file:

1. Random-access files can perform both input and output operations on an open file. Sequential files are opened for input or output but never both.

2. Random-access files can be arbitrarily positioned to locations within the file. Sequential files can be positioned to their origin via the REWD system call, but they cannot be positioned to other locations without reading or writing between the starting position and the desired position.

Random-access files are actually a special type of sequential file. The random-access file has a data structure written into it which facilitates positioning to arbitrary locations.

PHYSICAL AND LOGICAL RECORDS

There are two terms which must be differentiated in order to explain the functioning of random-access files. The first of these terms is physical record. A physical record is the block of data treated as a unit by the storage device being used.

In the case of floppy-disks, the physical record is also called sector because it is written (or read) out as a single unit. CP/68 allows the user to read and write arbitrary sectors with the IOHDR system call. Thus, random-access at the physical record level is provided in CP/68. The size of a physical record, however, is fixed by the hardware. This imposes severe restrictions on the user, whose data may not fit in the required record size. The user desires control over the size of record. It is desirable to vary the record size to fit the application. This variable-sized record is referred to as a logical record. The logical record does not depend on hardware; it is under program control. The

manipulation of logical records (hereafter simply called records) is done by the routines described in this manual. The routines in this package must convert the user's descriptions of logical records into internal descriptions in terms of physical records.

ENTRY POINTS IN THE RANDOM-ACCESS PACKAGE

There are seven entry points in this package.

1.	CREATE	build a new random-access file
2.	ROPEN	open an existing random-access file
3.	RCLOSE	close an open random-access file
4.	RREAD	read a byte from the current position of a random-access file
5.	RWRITE	put a byte into the current position of a random-access file
6.	POSITION	move the random-access file pointer to the start of a desired record.
7.	EXPAND	add new records to an open random-access file

User packages may link with these routines by using their names as EXTernals. Alternatively, a vector table is provided at the start of the random-access package which has jumps to each of the routines in the order given above. Each routine is called with the address of an FCB (File-control block) in the index register. The RREAD routine returns the byte just read in the A accumulator. The RWRITE routine is passed the byte to be written in the A accumulator.

THE RANDOM-ACCESS FILE-CONTROL BLOCK (FCB)

The file-control block (FCB) used with random files has five additional data fields appended to it, compared to the normal FCB as described in the CP/68 Advanced User's Guide. They are:

FCBRNM

This 2-byte field holds the number of records contained in the file. It must be set by the user when CREATE is called. It is set by the system on ROPEN. There is a maximum for this value, based on the sector size of the floppy disks in use, and given by the relation

MXRNUM = 20 * (SECSIZ-4)

where SECSIZ is the number of bytes in a disk sector. If SECSIZ=128, this value becomes 2480. For 256-byte sectors, the maximum is 5040 records.

FCBRSZ

This 2-byte field holds the number of bytes in each logical record. The user must set it when CREATE is called. It is set by the system on ROPEN. The record size can be as small as one byte or as large as 65535 (FFFF hex) bytes. It is recommended that record sizes be kept fairly large--there is a 3-byte overhead for each record in the file.

FCBRCD

This 2-byte field holds the record number representing the current file position. The system initializes it when the file is opened (the first record number is 1). The user must set this field before POSITION is called.

FCBPOS

This 2-byte field holds the present record pointer of the current file position. The system intializes it when the file is opened. It gives the location within the current record that data will be read from or written into. As data is read or written, FCBPOS is incremented until FCBRSZ is reached. At this point , FCBRCD is incremented and FCBPOS reinitialized. Thus, any byte in the file is addressed by its record pointer (FCBRCD) and its position within the record (FCBPOS).

FCBRTB

In order to rapidly address a record within a file, the random-ccess package builds a table of addresses at the time that the file is opened. This table is built in the FCB of the file and occupies 120 bytes. The table consists of a 2-byte entry for each sector of the random-access file index. Hence, the table supports up to 60 index sectors per file. This leads to the limitation on FCBRNM.

The following EQUates will address the new FCB fields when used like the EQUates defined for the other FCB fields.

```
FCBRNM EQU 42
FCBRSZ EQU 44
FCBRCD EQU 46
FCBPOS EQU 48
FCBRTB EQU 50
```

Note that the FCB for a random-access file must be 170 bytes long. (The sequential-file FCB required only 42 bytes).

DATA STRUCTURES IN RANDOM-ACCESS FILES

Every random-access file built by CP/68 contains a data structure termed an index. This index is itself a sequential file containing pointers to the data records contained in the file. Thus, each random-access file is two sequential files: an index and the data record.

The file's first four bytes contain the values of FCBRNM and FCBRSZ--which describe the size of the file and data records. The index follows these two values. This index consists of a 3-byte entry; the first byte represents the track on which the data record begins, the second represents the sector on which the data record begins, and the third byte represents the position of the record's first data byte within the sector. The pointers are written sequentially as their data records are allocated during the CREATE processing. The end of the index is marked by a pointer containing all zero values. The index is padded with nulls (zero values) to fill out the last sector.

Data records begin on the next sector of the random-access file. They are simply a sequence of bytes FCBRSZ long and initialized to zero during the CREATE processing. There are no end-of-record marks; the end of one record is contiguous with the start of the next sequential record. Reading or writing past the end of a data record will automatically spill over onto the next data record. The RREAD and RWRITE routines will update the pointers FCBRCD and FCBPOS to indicate the current file position. The POSITION routine can be called at any time to move the file pointers to the start of a desired record.

RANDOM-ACCESS FILE ROUTINES

CREATE

This routine builds the structures for a new random-access file on disk. The user must provide a random-access FCB (170 bytes long) with the drive, filename, record size, and number of records set up. (FCBGDT=DSK, FCBDRV, FCBNAM, FCBRSZ, FCBRNM) A new file will be created with an index for each record. Each record will be cleared to zero. The filetype of the file will be set to 02. All random-access files disable space-compression. CREATE is called with the address of the FCB in the index register. It returns status information in FCBSTA of the user FCB. CREATE destroys the contents of the A and B accumulators and the condition codes. It leaves the index register intact. A CREATEd file is not open--it must be opened by a call to ROPEN before it may be accessed. CREATE may take a long time to build a large random-access file, since it must write the index as well as each data record in the file.

ROPEN

This routine prepares a previously CREATEd file for use. It is called with the address of a user FCB in the index register. The drive and filename must be set up by the user (FCBGDT=DSK, FCBDRV, and FCBNAM). It reads FCBRNM and FCBRSZ from the file and places them in the user FCB (which must be 170 bytes long). It also stores the filetype (must be 02), access code, first track and sector (T/S), last T/S (FCBTYP, FCBACS, FCBFTS, and FCBLTS) fields into the FCB. The file pointers (FCBRCD and FCBPOS) are initialized to point to the first record in the file. The ROPEN routine also reads the file index, building a table (FCBRTB) in the FCB containing the track and sector of each sector of the index. All unused table entries are cleared. The process of building this table may take many seconds for a file with many data records. ROPEN destroys the A and B accumulators and the condition codes; it returns the index register intact. Error status is returned in the FCBSTA field of the user FCB.

RCLOSE

This routine closes the file described by the user FCB whose address is passed in the index register. Any pending output is completed before the FCB is de-allocated. RCLOSE should only be used on random-access files. (type=02) It destroys the A and B accumulators and condition codes; the index register is returned intact. Error status is returned in the FCBSTA field of the user FCB.

RREAD

This routine reads a data byte from a random-access file. It is called with the address of the user FCB in the index register. The data byte read is returned in the A accumulator. RREAD reads sequentially from the current file position defined by FCBRCD and FCBPOS. If the last operation performed on the file was writing, RREAD will finish that operation before reading. Subsequent calls to RREAD will access sequential data bytes. RREAD destroys the B accumulator and the condition codes; the index register is returned intact. Error status is returned in the FCBSTA field of the user FCB. If a read error occurs, RREAD will return a null.

RWRITE

This routine writes a data byte into a random-access file. It is called with the address of the user FCB in the index register and the byte to be written in the A accumulator. The data byte will be written at the current file position defined by FCBRCD and FCBPOS. Subsequent calls to RWRITE will write sequential data bytes. RWRITE destroys the A and B accumulators and the condition codes; the index register is returned intact. Error status is returned in the FCBSTA field of the user FCB.

POSITION

This routine moves the current file position to the start of a desired record in the file. It is called with the address of a user FCB in the index register. The desired record is set in the FCBRCD field of the FCB. POSITION will initialize FCBPOS when the desired record is found. If the last operation performed on the file was writing, the last write will be finished before the file position is changed. POSITION destroys the A and B accumulators and the condition codes; the index register is returned intact. Error status is returned in the FCBSTA field of the user FCB.

EXPAND

This routine adds new records to an existing, open, random-access file. EXPAND is called with the address of a user FCB in the index register. The number of new records desired is set in the FCBRCD field of the user FCB. The new records will have the same size (FCBRSZ) as the others in the file. EXPAND will close the file after the new records have been appended. None of the old records will be affected by the EXPAND process. The new records are added after all the old ones. A file may be EXPANDed many times. EXPAND destroys the A and B accumulators and the condition codes. The index register is returned intact. Error status is returned in the FCBSTA field of the user FCB. Adding many records to a file may take a long time.

NEW ERROR CODES FOR RANDOM-ACCESS FILES

The random-access routines trap all the same file errors as the sequential routines do. In addition, they trap four new errors that are specific to random-access operations. They are:

0B BAD RECORD SIZE PARAMETER
The value specified for FCBRSZ was zero.

0C BAD RECORD NUMBER PARAMETER
The value specified for FCBRNM was zero or greater

than the MXRNUM for the system sector size.

0E BAD FILE TYPE
The file specified is not random-access type. (02)

0F BAD POSITION PARAMETER
The value specified for FCBRCD lies outside the file. (The last data byte of the last data record has been written or read.)

NOTES AND WARNINGS

I.

Random-access files contain track/sector information in their indices. Hence, rearranging their sectors on the disk will corrupt the indexing and destroy the file. Disks which have random-access files on them should not be copied using the packing (drive:=drive:*.*) PIP command. Such disks should be copied exactly, sector-for-sector, using the nonpacking PIP copy command. (drive:=drive:) Using PIP to transfer a random-access file from disk to disk will corrupt the new file, making it worthless.

II.

The FCBDTT field of the FCB, which was used in sequential file handling to specify input or output, is under system control when working with random-access files. It should not be used by the programmer.

EXAMPLE OF THE USE OF CP/68 RANDOM-ACCESS FILE ROUTINES

The following program illustrates the use of random-access file routines under CP/68. It allows exercise of all the CP/68 random-file operations.

```
       NAM TESTRND
*
* EXERCISE PROGRAM FOR RANDOM-ACCESS FILES IN CP/68
*
* 'O'  OPEN FILE (ONLY ONE FILE OPEN AT A TIME)
* 'C'  CLOSE FILE
* 'B'  BUILD A NEW RANDOM-ACCESS FILE
* 'R'  READ FROM CURRENT POSITION IN FILE
*      (END ON CARRIAGE-RETURN IN FILE)
* 'W'  WRITE TO FILE AT CURRENT POSITION
*      (END WITH CARRIAGE-RETURN)
* 'P'  POSITION FILE TO DESIRED RECORD
* 'E'  EXPAND CURRENTLY-OPEN FILE
*
       JMP START
*
* DEFINE RANDOM-FILE EXTERNALS
*
       EXT CREATE
       EXT ROPEN
       EXT RCLOSE
       EXT RREAD
       EXT RWRITE
       EXT POSITION
       EXT EXPAND
*
* DEFINE TEXT BUFFER FOR OUTPUT
*
BUFFER RMB 80        80 CHARACTERS FOR LINE BUFFER
BUFEND FCB $0D       FORCE C.R. ON LINE
BUFPNT RMB 2         BUFFER POINTER STORAGE
*
* DEFINE CP/68 EQUATES
*
       BASEQU
       FCBDEF
FCBRNM EQU 42
FCBRSZ EQU 44
FCBRCD EQU 46
FCBPOS EQU 48
*
* LOCAL RANDOM-FCB BLOCK
*
FCBLK  RMB 2
```

```
        FCC 'DSK'
        RMB 2
        FDB SECBUF
        RMB 162
SECBUF  RMB 256
*
* SET OF PROGRAM PROMPT AND ERROR MESSAGES
*
M1      FCC 'ENTER COMMAND: '
        FCB 4
M2      FCC 'ENTER FILE SPECIFICATION: '
        FCB 4
M3      FCC 'ENTER RECORD SIZE : '
        FCB 4
M4      FCC 'ENTER NO. OF RECORDS: '
        FCB 4
M5      FCC 'ENTER RECORD NUMBER: '
        FCB 4
M6      FCC 'ENTER DATA: '
        FCB 4
M7      FCC 'BAD NUMBER'
        FCB $0D
*
*
* BEGIN PROGRAM CODE HERE
*
START   LDX #M1                 PROMPT FOR COMMAND
        PRTMSG
        GTCMD                   GET COMMAND
        LDX DESCRA
        LDA A 0,X
        CMP A #'O               "OPEN"?
        BNE NEX1                NO
*
* PROCESS "OPEN" COMMAND
*
        LDX #M2                 PROMPT FOR FILESPEC
        PRTMSG
        GTCMD                   GET FILESPEC.
        LDX DESCRA
        STX CUCHAR              BACK UP A TOKEN
        LDX #FCBLK
        FMTFCB                  PUT FILESPEC INTO FCB
        TST FCBSTA,X            ERROR?
        BEQ OPN2                NO
*
ERROR   LDX #FCBLK
        PRTERR                  PRINT ERROR MESSAGE (IF ANY)
        BRA START               GET NEW COMMAND
*
OPN2    JSR ROPEN               OPEN FILE
        BRA ERROR               ERROR (IF ANY) AND LOOP
```

```
*
NEX1    CMP A #'C           "CLOSE"?
        BNE NEX2            NO
*
* PROCESS "CLOSE" COMMAND
*
        LDX #FCBLK
        JSR RCLOSE          CLOSE FILE
        BRA ERROR           ERROR (IF ANY) AND LOOP
*
NEX2    CMP A #'B           "BUILD"?
        BNE NEX3            NO
*
* PROCESS "BUILD" COMMAND
*
        LDX #M2             PROMPT FOR FILESPEC
        PRTMSG
        GTCMD               GET FILESPEC.
        LDX DESCRA
        STX CUCHAR          BACK UP A TOKEN
        LDX #FCBLK
        FMTFCB              PUT FILESPEC INTO FCB
        TST FCBSTA,X        ERROR?
        BNE ERROR           YES
*
        LDX #M3             PROMPT FOR RECORD SIZE
        PRTMSG
        GTCMD               GET VALUE
        LDA B RC            NUMERIC?
        CMP B #3
        BEQ BLD2            YES
*
NUMERR  LDX #M7             PRINT "BAD NUMBER" MESSAGE
        PRTMSG
        JMP START           TRY AGAIN
*
BLD2    LDA A VALUE
        LDA B VALUE+1
        LDX #FCBLK          PUT RECSIZ INTO FCB
        STA A FCBRSZ,X
        STA B FCBRSZ+1,X
        LDX #M4             PROMPT FOR NO. OF RECORDS
        PRTMSG
        GTCMD               GET VALUE
        LDA B RC            NUMERIC?
        CMP B #3
        BNE NUMERR          NO
*
        LDA A VALUE
        LDA B VALUE+1
        LDX #FCBLK          PUT RECNUM INTO FCB
        STA A FCBRNM,X
```

```
        STA B FCBRNM+1,X
        JSR CREATE          BUILD NEW FILE
        JMP ERROR           ERROR (IF ANY) AND LOOP
*
NEX3    CMP A #'R           "READ"?
        BNE NEX4            NO
*
* PROCESS "READ" COMMAND
*
RED1    LDX #BUFFER         INIT. OUTPUT BUFFER POINTER
        STX BUFPNT
*
RED2    LDX #FCBLK
        JSR RREAD           READ BYTE FROM FILE
        TST FCBSTA,X        ERROR?
        BEQ RED3            NO
*
        JMP ERROR           YES
*
RED3    LDX BUFPNT          GET BUFFER POINTER
        STA A 0,X           STORE CHARACTER IN BUFFER
        CMP A #$0D          CARRIAGE-RETURN?
        BEQ RED4            IF SO, FINISH UP
*
        INX
        STX BUFPNT          MOVE BUFFER POINTER
        CPX #BUFEND         AT END OF BUFFER?
        BNE RED2            NO, LOOP
*
        LDX #BUFFER
        PRTMSG              PRINT BUFFER CONTENTS
        BRA RED1            LOOP FOR NEW BUFFER
*
RED4    LDX #BUFFER
        PRTMSG              PRINT BUFFER CONTENTS
        JMP START
*
NEX4    CMP A #'W           "WRITE"?
        BNE NEX5            NO
*
* PROCESS "WRITE" COMMAND
*
        LDX #M6             PROMPT FOR DATA
        PRTMSG
        GTCMD               GET DATA LINE
        LDX DESCRA          POINT TO IT
WRIT1   LDA A 0,X           GET DATA BYTE
        LDX #FCBLK
        JSR RWRITE          WRITE DATA BYTE
        TST FCBSTA,X        ERROR?
        BEQ WRIT2           NO
*
```

```
WRIT1A JMP ERROR            YES
*
WRIT2  LDX DESCRA
       LDA A 0,X            GET BYTE AGAIN
       CMP A #$0D           C.R.?
       BEQ WRIT1A           IF SO, DONE
*
       INX                  IF NOT, MOVE POINTER
       STX DESCRA
       BRA WRIT1            LOOP
*
NEX5   CMP A #'P            "POSITION"?
       BNE NEX6             NO
*
* PROCESS "POSITION" COMMAND
*
       LDX #M5              PROMPT FOR RECORD NUMBER
       PRTMSG
       GTCMD                GET VALUE
       LDA B RC             NUMERIC?
       CMP B #3
       BEQ POS1             YES
*
       JMP NUMERR           NO
*
POS1   LDA A VALUE
       LDA B VALUE+1
       LDX #FCBLK           PUT RECNUM INTO FCB
       STA A FCBRCD,X
       STA B FCBRCD+1,X
       JSR POSITION         POSITION FILE
       JMP ERROR            ERROR (IF ANY) AND LOOP
*
NEX6   CMP A #'E            "EXPAND"?
       BNE NEX7             NO
*
* PROCESS "EXPAND" COMMAND
*
       LDX #M4              PROMPT FOR NO. OF RECORDS
       PRTMSG
       GTCMD                GET VALUE
       LDA B RC             NUMERIC?
       CMP B #3
       BEQ EXP1             YES
*
       JMP NUMERR           NO
*
EXP1   LDA A VALUE
       LDA B VALUE+1
       LDX #FCBLK           PUT RECNUM INTO FCB
       STA A FCBRCD,X
       STA B FCBRCD+1,X
```

```
        JSR EXPAND          ENLARGE FILE
        JMP ERROR           ERROR (IF ANY) AND LOOP
*
NEX7    JMP START           UNRECOGNIZED COMMAND
*
        END
```

RANDOM-ACCESS FILE SUPPORT FOR STRUBAL+ PROGRAMS

All functions of the random-access file package are available to the STRUBAL+ programmer through procedures built into the random-access file-driver program supplied with the random-access package. This file-driver program includes all the support necessary for sequential file I/O plus all the additional random-file commands. Some of the random-file operations share the same keywords with the sequential operations. The shared keywords are:

OPEN	open a file for use
CLOSE	close a file after use
READ	read data from a file
WRITE	write data into a file

The new set of keywords includes:

BUILD	create a new random-access file
DELETE	delete a file from the disk (random or sequential)
ENLARGE	add records to a random-access file
LOCATE	return the current file pointers of a random-access file
PLACE	position a random-access file to a given record

This set of keywords provides support for all file manipulations under CP/68.

The shared keywords READ and WRITE work the same way for sequential and random-access files. Data is moved sequentially starting with the current file position. The .EOF. and .ERR. functions are used in the same way with random-access files as they were with sequential files. The shared keyword CLOSE also works the same for both types of files in CP/68. The shared keyword OPEN has the same syntax for both types of files. If a random-access file is to be opened, append ';R' to the file specification instead of ';I' or ';O' used with sequential files. This identifies the file to be opened for random-access.

Only files built for random access can be used as random-access files. Sequential files cannot be manipulated using random-access statements. A file with a filetype of 02 is a random-access file. Random-access files may be built under STRUBAL+ control using the BUILD procedure. They may be positioned to any desired record using the PLACE procedure. The current values of the file pointers may be obtained using the LOCATE procedure. Finally, records may be added to an existing random-access file through the use of the ENLARGE procedure.

BUILD procedure

This procedure is used to create a new random-access file. Such a file is defined by its file specification (drive, name, and extension), a record count, and a record size. If FNAME is a string of characters containing a valid CP/68 file designation (which does not already exist on the disk), RECNO is an integer which contains the desired number of records to be in the file, and RECSIZ is an integer which contains the desired number of characters to be in each record, then the following procedure call will build the desired file.

```
CALL BUILD(RFCB,FNAME,RECNO,RECSIZ)
```

RFCB is the name of the user-supplied file-control block (FCB). The FCB must contain 426 bytes for systems whose sector size is 256, and 298 bytes for systems whose sector size is 128 bytes. The BUILD procedure may take substantial time for a large file. The file is closed upon return from BUILD.

ENLARGE procedure

This procedure is used to add new records to an existing random-access file. The file must be already open before ENLARGE is called. ENLARGE requires the address of the file FCB and the desired number of records to be added as parameters.

```
CALL ENLARGE(RFCB,RECNO)
```

The file is closed upon return from ENLARGE. The ENLARGE procedure may take substantial time if many records are added to the file.

LOCATE procedure

This procedure returns the current file pointers of a random-access file. There are two pointers: the current record RECNO, and the current position within the record BYTNO. (These correspond to FCBRCD and FCBPOS.) LOCATE is called with the address of the file FCB as a parameter.

```
CALL LOCATE(RFCB,RECNO,BYTNO)
```

It returns two integer values containing the pointer contents.

PLACE procedure

This procedure moves the file pointers of a random-access file to a user-specified record. PLACE requires the address of the file FCB as a parameter, as well as an integer containing the desired record number.

```
CALL PLACE(RFCB,RECNO)
```

PLACE always positions the file to the start of the desired record.

USING RANDOM-ACCESS FILES IN STRUBAL+

The following STRUBAL+ example program illustrates the use of the random-access procedures to exercise random-access files. The example is similar in function to the assembly-language example shown earlier.

```
* ILLUSTRATE USE OF RANDOM-ACCESS FILES THROUGH STRUBAL+
* ASSUME RANDOM-FILE PACKAGE AND DRIVERS LOADED
*
      DSTRING DATA(80),RFCB(426),FNAME(30),CMD(10),TMP(1)
      INTEGER RECNO,RECSIZ,BYTNO
*
      CALL INITIO
*
* AVAILABLE COMMANDS ARE:
*
* BUILD, CLOSE, ENLARGE, OPEN, POSITION, READ, WRITE
*
START  INPUT /,'ENTER COMMAND (B,C,E,O,P,R,W): ',%CMD
      XTRACT TMP=1,CMD
      STRING IF TMP .NE. 'O' THEN NEX1
*
* PROCESS "OPEN" COMMAND HERE
*
      INPUT /,'ENTER FILE SPECIFICATION: ',%FNAME
      STRING FNAME=FNAME,';R'
      OPEN (RFCB) FNAME
      GOTO START
*
NEX1   STRING IF TMP .NE. 'C' THEN NEX2
*
* PROCESS "CLOSE" COMMAND HERE
*
      CLOSE (RFCB)
      GOTO START
*
NEX2   STRING IF TMP .NE. 'B' THEN NEX3
*
* PROCESS "BUILD" COMMAND HERE
*
      INPUT /,'ENTER FILE SPECIFICATION: ',%FNAME
      INPUT /,'ENTER NUMBER OF RECORDS: ',RECNO
      INPUT /,'ENTER RECORD SIZE: ',RECSIZ
      CALL BUILD(RFCB,FNAME,RECNO,RECSIZ)
      GOTO START
*
```

```
NEX3    STRING IF TMP .NE. 'R' THEN NEX4
*
* PROCESS "READ" COMMAND HERE
*
        CALL LOCATE(RFCB,RECNO,BYTNO)
        PRINT /,[6],'RECORD=',RECNO,'    BYTE=',BYTNO
* PRINT CURRENT POINTERS BEFORE READING
        READ (RFCB) %DATA
        PRINT /,[72],%DATA
        GOTO START
*
NEX4    STRING IF TMP .NE. 'W' THEN NEX5
*
* PROCESS "WRITE" COMMAND HERE
*
        INPUT /,'ENTER DATA: ',%DATA
        WRITE (RFCB) %DATA
        GOTO START
*
NEX5    STRING IF TMP .NE. 'P' THEN NEX6
*
* PROCESS "POSITION" COMMAND HERE
*
        INPUT /,'ENTER RECORD NUMBER: ',RECNO
        CALL PLACE(RFCB,RECNO)
        GOTO START
*
NEX6    STRING IF TMP .NE. 'E' THEN START
*
* PROCESS "ENLARGE" COMMAND HERE
*
        INPUT /,'ENTER NUMBER OF RECORDS: ',RECNO
        CALL ENLARGE(RFCB,RECNO)
        GOTO START
*
        END
```

DELETING A FILE USING STRUBAL+

One additional procedure is contained in the new file driver program; this procedure allows STRUBAL+ programs to delete files from disk. Only unambiguous names can be used; no wildcards are allowed. The DELETE procedure requires an FCB in the user program. This FCB can be sized either for sequential files or random-access files. The file specification is passed as a string to the procedure.

```
CALL DELETE(RFCB,DNAME)
```

Care should be taken with this procedure, as once a file is deleted it is lost. There will be no prompting, unlike the DELETE command under CP/68.

MODIFICATIONS FOR DISK HARDWARE DIFFERENCES

CP/68 can be tailored for a wide variety of disk configurations. This section will describe the places which must be modified for most common hardware setups. There are three parameters which describe a disk to CP/68:

SECSIZ the number of bytes in a sector (128 assumed)
TRKSIZ the number of sectors in a track (18 assumed)
DSKSIZ the number of tracks on a disk (35 assumed)

In addition, CP/68 checks the number of drives. From 1 to 4 drives may be used. (CP/68 as described here assumes four drives.) More than four drives can be used if more space is allocated to the free-space pointer table (FRETAB) in the base-page. Two bytes are needed for each drive added.

SECSIZ

This parameter is the most important one, as it affects the buffer sizes for the sector buffers in the system. Sector buffers appear in:

```
CLI-       SAVFCB, SYSFCB, SUBFCB
BOOT-      BUFFER
DELETE-    SYSFCB
INIT-      FCBSPC
LINK-      SYSFCB
PIP-       INFCB, OUTFCB
SECURITY-  SYSFCB
RNDFILE-   RNDFCB
```

All sector buffers are sized for 128 bytes as shown. They could be enlarged to 256 bytes if necessary. Larger sectors would require extensive modification since byte counts are kept in 8-bit locations throughout CP/68.

SECSIZ also is used as a parameter in CP/68 to allow addressing of elements of a sector or to compute constants based on the sector size. Use of SECSIZ as a parameter appears in:

CLI, DREAD, Sequential File I/O, BOOT, INITIALIZE, LINK and RNDFILE

TRKSIZ

This parameter is used in the following routines:

```
CLI-      in subroutine SEMPTY
DREAD-    in subroutine GETDR
INIT-
PIP-      in subroutine DTDCPY
FORMAT-
```

The use of TRKSIZ in INITIALIZE includes the length of the sector-interleave table TBL. There must be a table entry for each sector on a disk track.

DSKSIZ

This parameter is used in the following routines:

```
INIT-
PIP-      in subroutine DTDCPY
FORMAT-
```

CP/68 assumes that all disks have the same DSKSIZ.

Number of Drives in System

This parameter appears in the following routines:

```
CLI-      at WARM3 (to initialize FRETAB)
          in subroutine INICMD
          in subroutine DIRCMD
          in subroutine FMTFCB

SFIO-     (mask off low 2 bits of drive number to access FRETAB)

DELETE-
INITIALIZE-
LINK-
PIP-
SECURITY-
FORMAT-
RNDFILE- (mask off low 2 bits of drive number to access FRETAB)
```

In all cases except SFIO and RNDFILE, the checks on drive number are used for error-detection only.

DISK HANDLING SOFTWARE

Any disk operating system like CP/68 must be modified for use on different hardware. The hardware-specific code is localized in the DRIVERS, BOOT, and FORMAT. The DRIVERS require initialize, sector-read, and sector-write capabilities for multiple drives. BOOT requires only initialize and sector-read from drive 0. FORMAT requires track-seek and track-write capability for multiple drives. Drivers for several common disk configurations are given here. They each perform the same functions--only one is needed for CP/68.

MODIFICATIONS FOR VARIOUS SYSTEM MONITOR ROMS

CP/68 makes no use of system monitor routines during its execution. As a result, any of the current "---BUG" monitors can be used with it. BIOS contains the addressing for the various I/O devices (EQTAB), which may need changing for different addressing of I/O devices. BIOS also contains an error trap for CP/68 calls (SWIs) that have an invalid function code. This trap should vector to the normal breakpoint entry in the monitor ROM. This vector is directed to E113 hex in SWIHDR. CLI also contains a vector to the monitor in its command table (CMDTAB). The EXIT command is vectored to the warm-start entry of the monitor ROM (the version shown goes to E0E3 hex). The BOOT transient contains an error trap which is jumped to in case of disk errors during boot. This vector is shown as E113 hex (like the one in BIOS).

One other modification will be necessary to use CP/68--point the SWI vector of the system to the SWIHDR entry. Some means must be found to force SWIs to be processed by SWIHDR. The BOOT process must set up the SWI vector, or else it must be set by code at the COLDST entry in CLI.

Part 7

Software Listings
******** ********

Resident
- BIOS.....103
- CLI.....117
- DIRECTORY.....140
- SFIO.....142
- ICOM driver.....150

Transients
- ASSIGN.....153
- BOOT.....156
- DELETE.....158
- INITIALIZE.....162
- LINK.....165
- PIP.....168
- SECURITY.....184
- SET.....187
- STATUS.....189
- RANDOM.....191

Hex dump of resident code.....206
Load map.....208
Hex dump of transients.....209

Southwest Technical Products drivers.....211
- BOOT.....215
- INITIALIZE.....218
- FORMATTER.....221

Smoke Signal Broadcasting drivers.....224
- BOOT.....227
- INITIALIZE.....229
- FORMATTER.....232

Percom Data Company
Single-sector read and write.....236
- INITIALIZE.....238
- BOOT.....241

```
0001   0000 0000    N         NAM BIOS
0002                 *
0003                 *
0004                 *       COPYRIGHT 1978 BY HEMENWAY ASSOCIATES INC .
0005                 *       BOSTON MASS. 02111
0006                 *       ALL RIGHTS RESERVED
0007                 *
0008                 * RCB  EQU'S
0009                 *
0010   0000 0000     RCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0011   0000 0002     RCBGDT EQU 2          GENERIC DEVICE TYPE
0012   0000 0005     RCBSTA EQU 5          STATUS
0013   0000 0006     RCBDTT EQU 6          DATA TRANSFER TYPE
0014   0000 0007     RCBDBA EQU 7          DATA BUFFER ADDRESS
0015                 *
0016                 * STATUS EQU'S*
0017                 *
0018   0000 0080     BUSY   EQU $80        RCB BUSY STATUS
0019   0000 007F     NTBUSY EQU $7F        NOT BUSY (ANDED WITH STATUS)
0020                 *
0021   0000 03C0    N         ENT EQTAB
0022   0000 03F0    N         ENT PDTAB
0023                 *
0024                           BASEQU
0025 +  0000 0020     DESCRA EQU $20  DESCRIPTOR ADDRESS(2)
0026 +  0000 0022     DESCRC EQU $22  DESCRIPTOR COUNT
0027 +  0000 0023     CUCHAR EQU $23  CURRENT CHAR (2)
0028 +  0000 0025     RC     EQU $25  TOKEN RETURN CODE
0029 +  0000 0026     CLASS  EQU $26  TOKEN CLASS
0030 +  0000 0027     VALUE  EQU $27  BIN VALUE/TRANSFER ADDRESS (2)
0031 +  0000 0029     FCBCHN EQU $29  TOP OF FCB CHAIN (2)
0032 +  0000 002B     FRETAB EQU $2B  DISK FREE SPACE POINTER (8)
0033 +  0000 0033     BMEM   EQU $33  START OF TRANSIENT AREA(2)
0034 +  0000 0035     EMEM   EQU $35  END OF TRANSIENT AREA (2)
0035 +  0000 0037     CMEM   EQU $37  NEXT AVAIL TRANSIENT AREA (2)
0036 +  0000 0039     BS     EQU $39  BACKSPACE CHAR
0037 +  0000 003A     DL     EQU $3A  DELETE LINE CHAR
0038 +  0000 003B     DP     EQU $3B  DEPTH; LINES/PAGE
0039 +  0000 003C     DPCNT  EQU $3C  DEPTH TEMP
0040 +  0000 003D     WD     EQU $3D  WIDTH; CHARS/LINE
0041 +  0000 003E     NL     EQU $3E  NULL COUNT
0042 +  0000 003F     TB     EQU $3F  TAB CHAR
0043 +  0000 0040     DX     EQU $40  DUPLEX; FF=H, 00=F
0044 +  0000 0041     EJ     EQU $41  EJECT COUNT
0045 +  0000 0042     PS     EQU $42  PAUSE; 00=YES
0046 +  0000 0043     ES     EQU $43  ESCAPE CHAR
0047 +  0000 0044     LDP    EQU $44  DEPTH LINES/PAGE
0048 +  0000 0045     LDPCNT EQU $45  DEPTH TEMP
0049 +  0000 0046     LWD    EQU $46  WIDTH CHARS/LINE
0050                 *
0051                 * COLDSTART ENTRY TO THIS LOCATION
0052   0000 7E 0000 X         EXT @CLDST
0053                 *
0054                 *
0055                 * SWIHDR  SWI INPUT TO CP/68
0056                 * THIS IS THE ONLY ENTRY POINT TO BIOS
0057                 *
0058                 *
0059   0003 30       SWIHDR TSX            GET SP
0060   0004 EE 05             LDX 5,X       PT TO REQUEST CODE
0061   0006 4F                CLR A
0062   0007 E6 00             LDA B 0,X     GET REQUEST CODE
0063   0009 C1 35             CMP B #53     VALID RC?
0064   000B 23 03             BLS *+5       YES
0065                 *
0066   000D 7E E113           JMP $E113     NO
0067                 *
0068   0010 58                ASL B         *2
0069   0011 49                ROL A
0070                 *
0071                 *
0072                 * CALCULATE THE ADDRESS OF THE
0073                 * APPROPRIATE PROCESSING ROUTINE
0074                 *
0075   0012 8D 00             BSR START
0076                 *
0077   0014 30       START  TSX
0078   0015 EB 01             ADD B 1,X
0079   0017 A9 00             ADC A 0,X
0080                 *
0081   0019 CB 40             ADD B #DSPTAB
0082   001B 89 03             ADC A #DSPTAB/256
0083                 *
0084   001D A7 00             STA A 0,X
0085   001F E7 01             STA B 1,X
0086   0021 EE 00             LDX 0,X
0087   0023 EB 01             ADD B 1,X
0088   0025 A9 00             ADC A 0,X
0089                 *
0090   0027 30                TSX
0091   0028 A7 00             STA A 0,X
0092   002A E7 01             STA B 1,X
0093                 *
0094   002C A6 02             LDA A 2,X
0095   002E 06                TAP
0096   002F EE 00             LDX 0,X
0097                 *
0098   0031 31                INS
0099   0032 31                INS
0100                 *
0101   0033 AD 00             JSR 0,X       CALL PROCESSING ROUTINE
0102                 *
0103   0035 30                TSX
0104   0036 6C 06             INC 6,X
0105   0038 26 02             BNE *+4
0106                 *
0107   003A 6C 05             INC 5,X
0108   003C 3B                RTI           RETURN TO USER
0110                 * USER REGISTER EQUATES:
0111                 *
0112   003D 0002     UC     EQU 2
0113   003D 0003     UB     EQU 3
0114   003D 0004     UA     EQU 4
0115   003D 0005     UXH    EQU 5
0116   003D 0006     UXL    EQU 6
0117   003D 0007     URH    EQU 7
0118   003D 0008     URL    EQU 8
0119                 *
0120                 * PROCESSING ROUTINES:
0121                 *
0122                 * PSHALL: PUSHALL REGISTERS
```

```
0123                    *
0124   003D 34          @PSHAL DES
0125   003E 34                 DES
0126   003F 34                 DES
0127   0040 34                 DES
0128   0041 34                 DES
0129                    *
0130   0042 C6 09               LDA B #9
0131   0044 30                  TSX
0132                    *
0133   0045 A6 05       PSHALS LDA A 5,X
0134   0047 A7 00              STA A 0,X
0135   0049 08                 INX
0136   004A 5A                 DEC B
0137   004B 26 F8              BNE PSHALS
0138                    *
0139   004D C6 05               LDA B #5
0140   004F 30                  TSX
0141                    *
0142   0050 A6 02       PSHALC LDA A UC,X
0143   0052 A7 09              STA A UC+7,X
0144   0054 08                 INX
0145   0055 5A                 DEC B
0146   0056 26 F8              BNE PSHALC
0147                    *
0148   0058 39                 RTS
0149                    *
0150                    *
0151                    * PULALL: PULL ALL REGISTERS
0152                    *
0153   0059 30          @PULAL TSX
0154   005A C6 05              LDA B #5
0155                    *
0156   005C A6 09       PULAC  LDA A UC+7,X
0157   005E A7 02              STA A UC,X
0158   0060 08                 INX
0159   0061 5A                 DEC B
0160   0062 26 F8              BNE PULAC
0161                    *
0162   0064 C6 09               LDA B #9
0163                    *
0164   0066 A6 03       PULALS LDA A URL-5,X
0165   0068 A7 08              STA A URL,X
0166   006A 09                 DEX
0167   006B 5A                 DEC B
0168   006C 26 F8              BNE PULALS
0169                    *
0170   006E 31                 INS
0171   006F 31                 INS
0172   0070 31                 INS
0173   0071 31                 INS
0174   0072 31                 INS
0175                    *
0176   0073 39                 RTS
0177                    *
0178                    *
0179                    * TXAB:  TRANSFER X TO A,B
0180                    *
0181   0074 30          @TXAB  TSX
0182   0075 A6 05              LDA A UXH,X
0183   0077 E6 06              LDA B UXL,X
0184                    *
0185   0079 A7 04       STAB   STA A UA,X
0186   007B E7 03              STA B UB,X
0187                    *
0188   007D 39                 RTS
0189                    *
0190                    *
0191                    * TABX:  TRANSFER A,B TO X
0192                    *
0193   007E 30          @TABX  TSX
0194   007F A6 04              LDA A UA,X
0195   0081 A7 05              STA A UXH,X
0196   0083 A6 03              LDA A UB,X
0197   0085 A7 06              STA A UXL,X
0198                    *
0199   0087 39                 RTS
0200                    *
0201                    *
0202                    * XABX:  EXCHANGE A,B AND X
0203                    *
0204   0088 30          @XABX  TSX
0205   0089 A6 05              LDA A UXH,X
0206   008B 36                 PSH A
0207   008C E6 06              LDA B UXL,X
0208   008E 8D EF              BSR @TABX+1
0209   0090 32                 PUL A
0210   0091 20 E6              BRA STAB
0211                    *
0212                    *
0213                    * PSHX:  PUSH X
0214                    *
0215   0093 34          @PSHX  DES
0216   0094 34                 DES
0217   0095 30                 TSX
0218   0096 86 09              LDA A #9
0219                    *
0220   0098 E6 02       PSHXA  LDA B 2,X
0221   009A E7 00              STA B 0,X
0222   009C 08                 INX
0223   009D 4A                 DEC A
0224   009E 26 F8              BNE PSHXA
0225                    *
0226   00A0 30                 TSX
0227   00A1 A6 05              LDA A UXH,X
0228   00A3 A7 09              STA A UXH+4,X
0229   00A5 A6 06              LDA A UXL,X
0230   00A7 A7 0A              STA A UXL+4,X
0231                    *
0232   00A9 39                 RTS
0233                    *
0234                    *
0235                    * PULX:  PULL X
0236                    *
0237   00AA 30          @PULX  TSX
0238   00AB A6 09              LDA A UXH+4,X
0239   00AD A7 05              STA A UXH,X
0240   00AF A6 0A              LDA A UXL+4,X
0241   00B1 A7 06              STA A UXL,X
0242                    *
0243   00B3 86 09              LDA A #9
0244                    *
```

```
0245  00B5 E6 08     PULXA  LDA B 8,X
0246  00B7 E7 0A            STA B 10,X
0247  00B9 09               DEX
0248  00BA 4A               DEC A
0249  00BB 26 F8            BNE PULXA
0250                 *
0251  00BD 31               INS
0252  00BE 31               INS
0253                 *
0254  00BF 39               RTS
0255                 *
0256                 *
0257                 * ADDXAB:  ADD X TO A,B
0258                 *
0259  00C0 30        @ADXAB TSX
0260  00C1 8D C6            BSR @XABX+1
0261  00C3 8D 03            BSR @ADABX+1
0262  00C5 8D C2            BSR @XABX+1
0263                 *
0264                 *
0265                 * ADDABX:  ADD A,B TO X
0266                 *
0267  00C7 30        @ADABX TSX
0268  00C8 A6 03            LDA A UB,X
0269  00CA E6 04            LDA B UA,X
0270                 *
0271  00CC AB 06     ADDAB  ADD A UXL,X
0272  00CE A7 06            STA A UXL,X
0273                 *
0274  00D0 E9 05            ADC B UXH,X
0275                 *
0276  00D2 07        STAUXH TPA
0277  00D3 E7 05            STA B UXH,X
0278  00D5 6D 06            TST UXL,X
0279                 *
0280  00D7 27 02     TESTZ  BEQ TESTZA
0281                 *
0282  00D9 84 FB            AND A #$FB
0283                 *
0284  00DB A7 02     TESTZA STA A UC,X
0285                 *
0286  00DD 39               RTS
0287                 *
0288                 *
0289                 * ADDAX:  ADD A TO X
0290                 *
0291  00DE 30        @ADDAX TSX
0292  00DF A6 04            LDA A UA,X
0293                 *
0294  00E1 C6 00     ADDZ   LDA B #00
0295  00E3 20 E7            BRA ADDAB
0296                 *
0297                 *
0298                 * ADDBX:  ADD B TO X
0299                 *
0300  00E5 30        @ADDBX TSX
0301  00E6 A6 03            LDA A UB,X
0302  00E8 20 F7            BRA ADDZ
0303                 *
0304                 *
0305                 * SUBXAB:  SUBTRACT X FORM A,B
0306                 *
0307  00EA 30        @SBXAB TSX
0308  00EB 8D 9C            BSR @XABX+1
0309  00ED 8D 03            BSR @SBABX+1
0310  00EF 20 98            BRA @XABX+1
0311                 *
0312                 *
0313                 * SUBABX:  SUBTRACT A,B FROM X
0314                 *
0315  00F1 30        @SBABX TSX
0316  00F2 E6 05            LDA B UXH,X
0317  00F4 A6 06            LDA A UXL,X
0318                 *
0319  00F6 A0 03            SUB A UB,X
0320  00F8 A7 06            STA A UXL,X
0321                 *
0322  00FA E2 04            SBC B UA,X
0323  00FC 20 D4            BRA STAUXH
0324                 *
0325                 *
0326                 * SUBAX:  SUBTRACT A FROM X
0327                 *
0328  00FE 30        @SUBAX TSX
0329  00FF E6 04            LDA B UA,X
0330                 *
0331  0101 A6 06     @SUB   LDA A UXL,X
0332  0103 10               SBA
0333  0104 A7 06            STA A UXL,X
0334                 *
0335  0106 E6 05            LDA B UXH,X
0336  0108 C2 00            SBC B #00
0337  010A 20 C6            BRA STAUXH
0338                 *
0339                 *
0340                 * SUBBX:  SUBTRACT B FROM X
0341                 *
0342  010C 30        @SUBBX TSX
0343  010D E6 03            LDA B UB,X
0344  010F 20 F0            BRA @SUB
0345                 *
0346                 *
0347                 * INDEX:  X:=X+A*B
0348                 *
0349  0111 8D 38     @INDEX BSR MPY8
0350  0113 37               PSH B
0351  0114 16               TAB
0352  0115 32               PUL A
0353  0116 30               TSX
0354  0117 20 B3            BRA ADDAB
0355                 *
0356                 *
0357                 * MUL8:  A,B:=A*B
0358                 *
0359  0119 8D 30     @MUL8  BSR MPY8
0360  011B 30               TSX
0361  011C E7 03            STA B UB,X
0362  011E A7 04            STA A UA,X
0363  0120 07               TPA
0364  0121 5D               TST B
0365                 *
0366  0122 20 B3     JMPTZ  BRA TESTZ
```

```
0367                    *
0368                    *
0369                    * MUL16:   A,B,X:=A,B*X
0370                    *
0371  0124 86 10        @MUL16 LDA A #16
0372  0126 36                  PSH A
0373  0127 30                  TSX
0374  0128 4F                  CLR A
0375  0129 5F                  CLR B
0376  012A 66 06               ROR UXH+1,X
0377  012C 66 07               ROR UXL+1,X
0378                    *
0379  012E 24 04        MUL16L BCC MUL16S
0380  0130 EB 04               ADD B UB+1,X
0381  0132 A9 05               ADC A UA+1,X
0382                    *
0383  0134 46           MUL16S ROR A
0384  0135 56                  ROR B
0385  0136 66 06               ROR UXH+1,X
0386  0138 66 07               ROR UXL+1,X
0387  013A 6A 00               DEC 0,X
0388  013C 26 F0               BNE MUL16L
0389                    *
0390  013E 31                  INS
0391  013F 30                  TSX
0392  0140 E7 03               STA B UB,X
0393  0142 A7 04               STA A UA,X
0394                    *
0395  0144 07                  TPA
0396  0145 EA 05               ORA B UXH,X
0397  0147 AA 06               ORA A UXL,X
0398                    *
0399  0149 20 D7               BRA JMPTZ
0400                    *
0401                    *
0402                    * MPY8:  A,B:=UA*UB
0403                    *
0404  014B 86 08        MPY8   LDA A #8
0405  014D 36                  PSH A
0406                    *
0407  014E 4F                  CLR A
0408  014F 30                  TSX
0409  0150 E6 06               LDA B UB+3,X
0410  0152 56                  ROR B
0411                    *
0412  0153 24 02        MPY8L  BCC MPY8S
0413                    *
0414  0155 AB 07               ADD A UA+3,X
0415                    *
0416  0157 46           MPY8S  ROR A
0417  0158 56                  ROR B
0418  0159 6A 00               DEC 0,X
0419  015B 26 F6               BNE MPY8L
0420                    *
0421  015D 31                  INS
0422  015E 39                  RTS
0423                    *
0424                    * UNSIGNED 16-BIT DIVISION
0425                    *    A,B / X
0426                    *    QUOTIENT IN A,B
0427                    *    REMAINDER IN X
0428                    *
0429  015F 30           @DIV16 TSX
0430  0160 A6 04               LDA A UA,X
0431  0162 E6 03               LDA B UB,X
0432  0164 EE 05               LDX   UXH,X    GET ARGUMENTS
0433  0166 37                  PSH B          STACK INPUTS
0434  0167 36                  PSH A
0435                           PSHX
0436 +  0168 3F                 SWI
0437 +  0169 05                 FCB 5
0438  016A 34                  DES
0439  016B 30                  TSX
0440  016C 86 01               LDA A #1
0441  016E 6D 01               TST 1,X
0442  0170 2B 0B               BMI DIV153
0443                    *
0444  0172 4C           DIV151 INC A
0445  0173 68 02               ASL 2,X
0446  0175 69 01               ROL 1,X
0447  0177 2B 04               BMI DIV153
0448                    *
0449  0179 81 11               CMP A #17
0450  017B 26 F5               BNE DIV151
0451                    *
0452  017D A7 00        DIV153 STA A 0,X
0453  017F A6 03               LDA A 3,X
0454  0181 E6 04               LDA B 4,X
0455  0183 6F 03               CLR 3,X
0456  0185 6F 04               CLR 4,X
0457  0187 E0 02        DIV163 SUB B 2,X
0458  0189 A2 01               SBC A 1,X
0459  018B 24 07               BCC DIV165
0460                    *
0461  018D EB 02               ADD B 2,X
0462  018F A9 01               ADC A 1,X
0463  0191 0C                  CLC
0464  0192 20 01               BRA DIV167
0465                    *
0466  0194 0D           DIV165 SEC
0467                    *
0468  0195 69 04        DIV167 ROL 4,X
0469  0197 69 03               ROL 3,X
0470  0199 64 01               LSR 1,X
0471  019B 66 02               ROR 2,X
0472  019D 6A 00               DEC 0,X
0473  019F 26 E6               BNE DIV163
0474                    *
0475  01A1 31                  INS            REPLACE STACK POSITION
0476  01A2 31                  INS
0477  01A3 31                  INS
0478  01A4 30                  TSX
0479  01A5 08                  INX            RECOVER POINTER POSITION
0480  01A6 08                  INX
0481  01A7 A7 05               STA A UXH,X    STORE REMAINDER
0482  01A9 E7 06               STA B UXL,X
0483  01AB 32                  PUL A
0484  01AC 33                  PUL B
0485  01AD A7 04               STA A UA,X     STORE QUOTIENT
0486  01AF E7 03               STA B UB,X
0487  01B1 07                  TPA
0488  01B2 5D                  TST B
```

```
0489  01B3 7E 00D7 R        JMP TESTZ      FIX UP ZERO FLAG
0490                  *
0491                  *
0492                  * MOVC:        FROM,TO ON STACK
0493                  *             B=COUNT
0494                  *             ON RETURN B=COUNT
0495                  *                       TO=TO+COUNT
0496                  *                       FROM=FROM+COUNT
0497                  *
0498  01B6 0009       FROM    EQU 9
0499  01B6 000B       TO      EQU 11
0500                  *
0501  01B6 30         @MOVC   TSX
0502  01B7 E6 03              LDA B UB,X     GET COUNT
0503                  *
0504  01B9 EE 09      MOVC1   LDX FROM,X
0505  01BB A6 00              LDA A 0,X      GET CHAR
0506                  *
0507  01BD 30                 TSX
0508  01BE EE 0B              LDX TO,X
0509  01C0 A7 00              STA A 0,X      MOVE CHAR
0510                  *
0511  01C2 30                 TSX
0512  01C3 5A                 DEC B          DONE?
0513  01C4 27 0E              BEQ MOVC3      YES
0514                  *
0515  01C6 6C 0A              INC FROM+1,X
0516  01C8 26 02              BNE MOVC2
0517                  *
0518  01CA 6C 09              INC FROM,X
0519                  *
0520  01CC 6C 0C      MOVC2   INC TO+1,X
0521  01CE 26 E9              BNE MOVC1
0522                  *
0523  01D0 6C 0B              INC TO,X
0524  01D2 20 E5              BRA MOVC1
0525                  *
0526  01D4 39         MOVC3   RTS
0527                  *
0528                  *
0529                  *
0530                  * CMPC:  COMPARE TWO STRINGS
0531                  *
0532                  *       PARM1,PARM2 ON STACK,  B=COUNT
0533                  *       ON MATCH PARMS=NEXT CHAR POSITION
0534                  *       ON MISMATCH PARMS=LAST CHAR POSITION
0535                  *
0536  01D5 0009       PARM1   EQU 9
0537  01D5 000B       PARM2   EQU 11
0538                  *
0539  01D5 30         @CMPC   TSX
0540  01D6 E6 03              LDA B UB,X
0541                  *
0542  01D8 EE 09      CMPO    LDX PARM1,X
0543  01DA A6 00              LDA A 0,X              GET A CHAR
0544  01DC 81 04              CMP A #$04             TERMINATOR?
0545  01DE 27 1F              BEQ CDONE              YES
0546                  *
0547  01E0 30                 TSX
0548  01E1 EE 0B              LDX PARM2,X
0549  01E3 37                 PSH B
0550  01E4 E6 00              LDA B 0,X
0551  01E6 C1 04              CMP B #$04             TERMINATOR?
0552  01E8 33                 PUL B
0553  01E9 27 14              BEQ CDONE              YES
0554                  *
0555  01EB A1 00              CMP A 0,X              MATCH?
0556  01ED 26 10              BNE CDONE              NO
0557                  *
0558  01EF 30                 TSX
0559  01F0 6C 0A              INC PARM1+1,X
0560  01F2 26 02              BNE CMP1
0561                  *
0562  01F4 6C 09              INC PARM1,X
0563                  *
0564  01F6 6C 0C      CMP1    INC PARM2+1,X
0565  01F8 26 02              BNE CMP2
0566                  *
0567  01FA 6C 0B              INC PARM2,X
0568                  *
0569  01FC 5A         CMP2    DEC B                  ALL DONE?
0570  01FD 26 D9              BNE CMPO               NO
0571                  *
0572  01FF 30         CDONE   TSX
0573  0200 07                 TPA
0574  0201 A7 02              STA A UC,X             SET CC
0575  0203 E7 03              STA B UB,X             SET B
0576                  *
0577  0205 39                 RTS
0578                  * CMWC:        COMPARE STRINGS WITH WILD CHARACTERS
0579                  *
0580                  *             SAME PARMS AS CMPC
0581                  *
0582  0206 30         @CMWC   TSX
0583  0207 E6 03              LDA B UB,X
0584                  *
0585  0209 EE 09      CMWO    LDX PARM1,X
0586  020B A6 00              LDA A 0,X
0587  020D 81 04              CMP A #4       TERMINATOR?
0588  020F 27 EE              BEQ CDONE
0589                  *
0590  0211 30                 TSX
0591  0212 EE 0B              LDX PARM2,X
0592  0214 37                 PSH B
0593  0215 E6 00              LDA B 0,X
0594  0217 C1 04              CMP B #4       TERMINATOR?
0595  0219 33                 PUL B
0596  021A 27 E3              BEQ CDONE      YES
0597                  *
0598  021C 81 3F              CMP A #'?      WILD CHARACTER?
0599  021E 27 04              BEQ CMW1       YES
0600                  *
0601  0220 A1 00              CMP A 0,X      MATCH?
0602  0222 26 DB              BNE CDONE      NO
0603                  *
0604  0224 30         CMW1    TSX
0605  0225 6C 0A              INC PARM1+1,X
0606  0227 26 02              BNE *+4
0607                  *
0608  0229 6C 09              INC PARM1,X
0609                  *
0610  022B 6C 0C              INC PARM2+1,X
```

```
0611  022D 26 02          BNE #+4
0612                *
0613  022F 6C 0B          INC PARM2,X
0614                *
0615  0231 5A             DEC B           DONE?
0616  0232 26 D5          BNE CMWO        NO
0617                *
0618  0234 20 C9          BRA CDONE       YES
0619                *
0620                * MOVS:       MOVE A VARIABLE LENGTH STRING TERMINATED (04)
0621                *
0622                *             FROM, TO ON STACK
0623                *             ON RETURN TO=TO+COUNT
0624                *                       FROM=FROM+COUNT
0625                *
0626                *
0627  0236 30       @MOVS  TSX
0628                *
0629  0237 EE 09    MOVS1  LDX FROM,X
0630  0239 A6 00           LDA A 0,X       GET CHARACTER
0631                *
0632  023B 30              TSX
0633  023C EE 0B           LDX TO,X
0634  023E A7 00           STA A 0,X       MOVE CHARACTER
0635                *
0636  0240 30              TSX
0637  0241 81 04           CMP A #04       DONE?
0638  0243 27 0E           BEQ MOVS3       YES
0639                *
0640  0245 6C 0A           INC FROM+1,X
0641  0247 26 02           BNE MOVS2
0642                *
0643  0249 6C 09           INC FROM,X
0644                *
0645  024B 6C 0C    MOVS2  INC TO+1,X
0646  024D 26 E8           BNE MOVS1
0647                *
0648  024F 6C 0B           INC TO,X
0649  0251 20 E4           BRA MOVS1
0650                *
0651  0253 39       MOVS3  RTS
0653                * FMTS:  REFORMAT A FILE NAME
0654                *
0655                *       FROM,TO ON STACK
0656                *       B=COUNT OF FROM STRING INCLUDING " "
0657                *
0658                *       B=RC= 00 UNAMBIG
0659                *             01 AMBIG
0660                *             02 BAD FILE NAME
0661                *
0662                *
0663                *
0664  0254 30       @FMTS  TSX
0665  0255 E6 03           LDA B UB,X              GET COUNT
0666  0257 E7 04           STA B UA,X              SAVE IN UA
0667  0259 C1 0C           CMP B #12               NAME TOO LONG?
0668  025B 23 06           BLS FMTSA               NO
0669                *
0670                * TOO MANY CHARACTERS
0671                *
0672  025D 30       FMTSO  TSX
0673  025E C6 02           LDA B #2                SET RC
0674  0260 E7 03           STA B UB,X                "
0675  0262 39              RTS
0676                *
0677                * FORMAT "TO" STRING
0678                *
0679  0263 EE 0B    FMTSA  LDX TO,X                POINT TO "TO"
0680  0265 86 20           LDA A #$20              BLANK
0681  0267 C6 08           LDA B #8
0682                *
0683  0269 A7 00    FMTSB  STA A 0,X               STORE BLANK
0684  026B 08              INX
0685  026C 5A              DEC B                   DONE?
0686  026D 26 FA           BNE FMTSB               NO
0687                *
0688  026F 86 2E           LDA A #'.
0689  0271 A7 00           STA A 0,X               STORE "."
0690  0273 08              INX
0691  0274 C6 03           LDA B #3
0692  0276 86 20           LDA A #$20              BLANK
0693                *
0694  0278 A7 00    FMTSC  STA A 0,X               STORE BLANK
0695  027A 08              INX
0696  027B 5A              DEC B                   DONE?
0697  027C 26 FA           BNE FMTSC               NO
0698                *
0699  027E 30       FMTS1  TSX
0700  027F 6F 03           CLR UB,X                SET DEFAULT RC
0701                *
0702                * FIND " "
0703                *
0704  0281 EE 09           LDX FROM,X              POINT TO FROM STRING
0705  0283 5F              CLR B                   CLEAR COUNT
0706                *
0707  0284 A6 00    FMTS2  LDA A 0,X               GET CHARACTER
0708  0286 81 2E           CMP A #'.
0709  0288 27 08           BEQ FMTS3               FOUND
0710                *
0711  028A 08              INX
0712  028B 5C              INC B
0713  028C C1 09           CMP B #9                NAME TO LONG?
0714  028E 26 F4           BNE FMTS2               NO
0715                *
0716  0290 20 CB           BRA FMTSO               YES FORMAT ERROR
0717                *
0718                * FOUND "." CHECK EXT
0719                *
0720  0292 5C       FMTS3  INC B
0721  0293 30              TSX
0722  0294 A6 04           LDA A UA,X              GET COUNT
0723  0296 10              SBA                     GET EXT COUNT
0724  0297 A7 04           STA A UA,X              SAVE
0725  0299 27 C2           BEQ FMTSO               NO EXT
0726                *
0727  029B 81 03           CMP A #3                TOO LONG?
0728  029D 22 BE           BHI FMTSO               YES
0729                *
0730                * EXT FIELD OK!
0731                *
0732  029F EE 09           LDX FROM,X              POINT TO FROM
0733  02A1 A6 00           LDA A 0,X               GET FIRST CHAR OF NAME
```

```
0734  02A3 81 2E          CMP A #'.           NO NAME?
0735  02A5 27 B6          BEQ FMTS0           YES
0736                *
0737  02A7 81 2A          CMP A #'*           WILD CARD?
0738  02A9 26 2C          BNE FMTS5           NO
0739                *
0740                * WILD CARD FILL WITH "?"
0741                *
0742  02AB 30             TSX
0743  02AC EE 0B          LDX TO,X            POINT TO "TO" STRING
0744  02AE C6 08          LDA B #8
0745  02B0 86 3F          LDA A #'?
0746                *
0747  02B2 A7 00    FMTS4 STA A 0,X           STORE "?"
0748  02B4 08             INX
0749  02B5 5A             DEC B               DONE?
0750  02B6 26 FA          BNE FMTS4           NO
0751                *
0752                * SET AMBIG RC
0753                *
0754  02B8 30             TSX
0755  02B9 86 01          LDA A #1            RC
0756  02BB A7 03          STA A UB,X          STORE
0757                *
0758                * FIX POINTERS
0759                *
0760  02BD A6 0A          LDA A FROM+1,X
0761  02BF 8B 02          ADD A #2
0762  02C1 A7 0A          STA A FROM+1,X
0763                *
0764  02C3 A6 09          LDA A FROM,X
0765  02C5 89 00          ADC A #00
0766  02C7 A7 09          STA A FROM,X
0767                *
0768  02C9 A6 0C          LDA A TO+1,X
0769  02CB 8B 08          ADD A #8
0770  02CD A7 0C          STA A TO+1,X
0771                *
0772  02CF A6 0B          LDA A TO,X
0773  02D1 89 00          ADC A #00
0774  02D3 A7 0B          STA A TO,X
0775                *
0776  02D5 20 34          BRA FMTS6
0777                *
0778                * MOVE NAME FROM -> TO
0779                *
0780  02D7 C6 08    FMTS5 LDA B #8
0781                *
0782  02D9 30       FMTS5A TSX
0783  02DA EE 09          LDX FROM,X
0784  02DC A6 00          LDA A 0,X           GET CHARACTER
0785  02DE 30             TSX
0786  02DF 6C 0A          INC FROM+1,X
0787  02E1 26 02          BNE *+4
0788                *
0789  02E3 6C 09          INC FROM,X
0790                *
0791  02E5 EE 0B          LDX TO,X
0792  02E7 81 2E          CMP A #'            DONE?
0793  02E9 27 15          BEQ FMTS5B          YES
0794                *
0795  02EB A7 00          STA A 0,X           STORE CHAR
0796  02ED 30             TSX
0797  02EE 5A             DEC B
0798  02EF 6C 0C          INC TO+1,X
0799  02F1 26 02          BNE *+4
0800                *
0801  02F3 6C 0B          INC TO,X
0802                *
0803  02F5 81 3F          CMP A #'?           WC?
0804  02F7 26 E0          BNE FMTS5A          NO
0805                *
0806  02F9 86 01          LDA A #1            SET AMBIG RC
0807  02FB 30             TSX
0808  02FC A7 03          STA A UB,X
0809  02FE 20 D9          BRA FMTS5A
0810                *
0811  0300 30       FMTS5B TSX                FIX "TO" POINTER
0812  0301 EB 0C          ADD B TO+1,X
0813  0303 E7 0C          STA B TO+1,X
0814  0305 E6 0B          LDA B TO,X
0815  0307 C9 00          ADC B #00
0816  0309 E7 0B          STA B TO,X
0817                *
0818                *
0819                *
0820                * FROM -> EXT
0821                * TO   ->
0822                *
0823                * PROCESS EXT
0824                *
0825  030B 30       FMTS6 TSX
0826  030C EE 09          LDX FROM,X
0827  030E A6 00          LDA A 0,X
0828  0310 81 2A          CMP A #'*           WILD CARD?
0829  0312 26 14          BNE FMTS8           NO
0830                *
0831                * YES FILL EXT WITH "?"
0832                *
0833  0314 30             TSX
0834  0315 EE 0B          LDX TO,X
0835  0317 08             INX                 SKIP OVER "."
0836  0318 C6 03          LDA B #3
0837  031A 86 3F          LDA A #'?
0838                *
0839  031C A7 00    FMTS7 STA A 0,X           STORE "?"
0840  031E 08             INX
0841  031F 5A             DEC B
0842  0320 26 FA          BNE FMTS7
0843                *
0844                * SET AMBIG RC
0845                *
0846  0322 30             TSX
0847  0323 86 01          LDA A #1
0848  0325 A7 03          STA A UB,X
0849  0327 39             RTS                 ALL DONE
0850                *
0851                *
0852                * MOVE "FROM" EXT TO "TO" EXT
0853                *
0854  0328 30       FMTS8 TSX
0855  0329 6C 0C          INC TO+1,X          FIX TO POINTER
```

```
0856  032B 26 02          BNE *+4
0857                      *
0858  032D 6C 0B          INC TO,X
0859                      *
0860  032F E6 04          LDA B UA,X          GET EXT COUNT
0861                      *
0862  0331 30       FMTS9 TSX
0863  0332 EE 09          LDX FROM,X
0864  0334 A6 00          LDA A 0,X
0865                      *
0866  0336 30             TSX
0867  0337 6C 0A          INC FROM+1,X
0868  0339 26 02          BNE *+4
0869                      *
0870  033B 6C 09          INC FROM,X
0871                      *
0872  033D EE 0B          LDX TO,X
0873  033F A7 00          STA A 0,X
0874                      *
0875  0341 30             TSX
0876  0342 6C 0C          INC TO+1,X
0877  0344 26 02          BNE *+4
0878                      *
0879  0346 6C 0B          INC TO,X
0880                      *
0881  0348 81 3F          CMP A #'?           WC?
0882  034A 26 04          BNE FMTS10          NO
0883                      *
0884  034C 86 01          LDA A #1            YES SET AMBIG RC
0885  034E A7 03          STA A UB,X
0886                      *
0887  0350 5A       FMTS10 DEC B
0888  0351 26 DE          BNE FMTS9
0889                      *
0890  0353 39             RTS                 ALL DONE
0892                      * RELATIVE ENTRY TABLE TO PROCESSING ROUTINES
0893                      *
0894  0354 0340   DSPTAB EQU *-START
0895                      *
0896                      *
0897  0354 FCE9           FDB *-@PSHAL*$FFFF   0
0898  0356 FD03           FDB *-@PULAL*$FFFF   1
0899  0358 FD1C           FDB *-@TXAB*$FFFF    2
0900  035A FD24           FDB *-@TABX*$FFFF    3
0901  035C FD2C           FDB *-@XABX*$FFFF    4
0902  035E FD35           FDB *-@PSHX*$FFFF    5
0903  0360 FD4A           FDB *-@PULX*$FFFF    6
0904  0362 FD5E           FDB *-@ADXAB*$FFFF   7
0905  0364 FD63           FDB *-@ADABX*$FFFF   8
0906  0366 FD78           FDB *-@ADDAX*$FFFF   9
0907  0368 FD7D           FDB *-@ADDBX*$FFFF   10
0908  036A FD80           FDB *-@SBXAB*$FFFF   11
0909  036C FD85           FDB *-@SBABX*$FFFF   12
0910  036E FD90           FDB *-@SUBAX*$FFFF   13
0911  0370 FD9C           FDB *-@SUBBX*$FFFF   14
0912  0372 FDA7           FDB *-@MUL8*$FFFF    15
0913  0374 FDB0           FDB *-@MUL16*$FFFF   16
0914  0376 FE40           FDB *-@MOVC*$FFFF    17
0915  0378 FE5D           FDB *-@CMPC*$FFFF    18
0916  037A 00AF           FDB *-@IOHDR*$FFFF   19
0917  037C 02EE           FDB *-@OPEN*$FFFF    20
0918  037E 02EF           FDB *-@CLOSE*$FFFF   21
0919  0380 02F0           FDB *-@REWD*$FFFF    22
0920  0382 02F1           FDB *-@OPEND*$FFFF   23
0921  0384 02F2           FDB *-@READ*$FFFF    24
0922  0386 02F3           FDB *-@WRITE*$FFFF   25
0923  0388 02F4           FDB *-@GETDR*$FFFF   26
0924  038A 02F5           FDB *-@PUTDR*$FFFF   27
0925  038C 02F6           FDB *-@DELET*$FFFF   28
0926  038E 02F7           FDB *-@CHAIN*$FFFF   29
0927  0390 02F8           FDB *-@PRTERR*$FFFF  30
0928  0392 02F9           FDB *-@WRMST*$FFFF   31
0929  0394 02FA           FDB *-@USR6*$FFFF    32
0930  0396 02FB           FDB *-@USR7*$FFFF    33
0931  0398 02FC           FDB *-@USR8*$FFFF    34
0932  039A 02FD           FDB *-@USR9*$FFFF    35
0933  039C 02FE           FDB *-@USR10*$FFFF   36
0934  039E 02FF           FDB *-@LOADB*$FFFF   37
0935  03A0 0300           FDB *-@LOADR*$FFFF   38
0936  03A2 030D           FDB *-@USR1*$FFFF    39
0937  03A4 030E           FDB *-@USR2*$FFFF    40
0938  03A6 030F           FDB *-@USR3*$FFFF    41
0939  03A8 0310           FDB *-@USR4*$FFFF    42
0940  03AA 0311           FDB *-@USR5*$FFFF    43
0941  03AC 0312           FDB *-@FMTFCB*$FFFF  44
0942  03AE FE88           FDB *-@MOVS*$FFFF    45
0943  03B0 FD61           FDB *-@INDEX*$FFFF   46
0944  03B2 02F1           FDB *-@NXTOK*$FFFF   47
0945  03B4 02F2           FDB *-@GTCMD*$FFFF   48
0946  03B6 02F3           FDB *-@PRTMSG*$FFFF  49
0947  03B8 FDA7           FDB *-@DIV16*$FFFF   50
0948  03BA 02F2           FDB *-@INTDK*$FFFF   51
0949  03BC FE98           FDB *-@FMTS*$FFFF    52
0950  03BE FE48           FDB *-@CMWC*$FFFF    53
0952                        *       EQUIPMENT TABLE:
0953                        *
0954  03C0 03C0     R EQTAB  EQU *
0955                        *
0956  03C0 049C     R CONSOL FDB INLIN
0957  03C2 04E7     R        FDB OTLIN
0958  03C4 8008              FDB $8008
0959                        *
0960  03C6 0593     R PTRDR  FDB INRDR
0961  03C8 0499     R        FDB NULL
0962  03CA 8010              FDB $8010
0963                        *
0964  03CC 0499     R PTPCH  FDB NULL
0965  03CE 05CB     R        FDB OTPCH
0966  03D0 8010              FDB $8010
0967                        *
0968  03D2 065E     R DISK   FDB .RDSEC
0969  03D4 0661     R        FDB .WTSEC
0970  03D6 0000              FDB 0
0971                        *
0972  03D8 0499     R LPTR   FDB NULL
0973  03DA 0603     R        FDB OTLPT
0974  03DC 8002              FDB $8002
0975                        *
0976  03DE 0664     R MTAPE  FDB .MTIN
0977  03E0 0667     R        FDB .MTOT
0978  03E2 0000              FDB 0
0979                        *
```

```
0980   03E4 049C     R TTYIO  FDB INLIN
0981   03E6 04E7     R        FDB OTLIN
0982   03E8 8010              FDB $8010
0983                          *
0984   03EA 0499     R NULLIO FDB NULL
0985   03EC 0499     R        FDB NULL
0986   03EE 0000              FDB 0000
0988                          * PHYSICAL DEVICE TABLE:
0989                          *
0990   03F0 43        PDTAB   FCC 'CON'
0991   03F3 03C0     R        FDB CONSOL
0992   03F5 03C0     R        FDB CONSOL
0993                          *
0994   03F7 50                FCC 'PTR'
0995   03FA 03C6     R        FDB PTRDR
0996   03FC 03C6     R        FDB PTRDR
0997                          *
0998   03FE 50                FCC 'PTP'
0999   0401 03CC     R        FDB PTPCH
1000   0403 03CC     R        FDB PTPCH
1001                          *
1002   0405 44                FCC 'DSK'
1003   0408 03D2     R        FDB DISK
1004   040A 03D2     R        FDB DISK
1005                          *
1006   040C 4C                FCC 'LPT'
1007   040F 03D8     R        FDB LPTR
1008   0411 03D8     R        FDB LPTR
1009                          *
1010   0413 4D                FCC 'MTA'
1011   0416 03DE     R        FDB MTAPE
1012   0418 03DE     R        FDB MTAPE
1013                          *
1014   041A 54                FCC 'TTY'
1015   041D 03E4     R        FDB TTYIO
1016   041F 03E4     R        FDB TTYIO
1017                          *
1018   0421 4E                FCC 'NUL'
1019   0424 03EA     R        FDB NULLIO
1020   0426 03EA     R        FDB NULLIO
1021                          *
1022   0428 00                FCB 0
1024                          * IOHDR: I/O HANDLER
1025                          *
1026   0429 30        @IOHDR TSX
1027   042A EE 05             LDX UXH,X              GET RCB ADDR
1028   042C 86 80             LDA A #BUSY
1029   042E A7 05             STA A RCBSTA,X         INIT BUSY BIT
1030                          *
1031                          * SEARCH PDTAB FOR EQT
1032                          *
1033   0430 BD 0440 R         JSR PDSRCH
1034   0433 25 04             BCS IOHDRA             NOT FOUND
1035                          *
1036                          * FOUND ENTRY; X=RCBADR; A,B=DRIVER ADR
1037                          *
1038                          XABX            X:=DRIVERADR; A,B:=RCBADR
1039 +  0435 3F               SWI
1040 +  0436 04               FCB 4
1041                          *
1042   0437 AD 00             JSR 0,X         CALL DRIVER
1043                          *
1044                          * ON RETURN X=RCBADR
1045                          *
1046   0439 86 7F     IOHDRA  LDA A #NTBUSY
1047   043B A4 05             AND A RCBSTA,X
1048   043D A7 05             STA A RCBSTA,X         TURN OFF BUSY BIT
1049                          *
1050   043F 39                RTS
1052                          * SEARCH PDTAB FOR DEVICE
1053                          *
1054                          * X=RCBADR
1055                          *
1056                          PDSRCH PSHX                    SAVE RCBADR
1057 +  0440 3F                SWI
1058 +  0441 05                FCB 5
1059   0442 08                INX                    PT TO RCBGDT
1060   0443 08                INX
1061                          *
1062                          PSHX                   SAVE PTR TO RCBGDT
1063 +  0444 3F                SWI
1064 +  0445 05                FCB 5
1065                          *
1066   0446 CE 03F0 R         LDX #PDTAB             POINT TO PDTAB
1067                          *
1068                          PDSRCA PSHX                    SAVE ON STACK
1069 +  0449 3F                SWI
1070 +  044A 05                FCB 5
1071   044B C6 03             LDA B #3               CHARS TO MATCH
1072                          CMPC
1073 +  044D 3F                SWI
1074 +  044E 12                FCB 18
1075   044F 27 26             BEQ PDSRCB             MATCH!
1076                          *
1077                          * NOMATCH
1078                          *
1079                          PULX                   GET PD PTR
1080 +  0451 3F                SWI
1081 +  0452 06                FCB 6
1082                          ADDBX                  POINT TO NEXT ENTRY
1083 +  0453 3F                SWI
1084 +  0454 0A                FCB 10
1085   0455 08                INX
1086   0456 08                INX
1087   0457 08                INX
1088   0458 08                INX
1089                          TXAB                   SAVE PD-PTR
1090 +  0459 3F                SWI
1091 +  045A 02                FCB 2
1092                          PULX                   SKIP RCBGDT
1093 +  045B 3F                SWI
1094 +  045C 06                FCB 6
1095                          PULX                   GET RCBADR
1096 +  045D 3F                SWI
1097 +  045E 06                FCB 6
1098                          PSHX                   SAVE
1099 +  045F 3F                SWI
1100 +  0460 05                FCB 5
1101   0461 08                INX
1102   0462 08                INX
1103                          PSHX                   SAVE RCBGDT
1104 +  0463 3F                SWI
```

```
1105 +  0464 05                FCB 5
1106                           TABX                    RESTORE PD-PTR
1107 +  0465 3F                SWI
1108 +  0466 03                FCB 3
1109    0467 6D 00             TST 0,X                 END OF TABLE?
1110    0469 26 DE             BNE PDSRCA              NO
1111                   *
1112                   * NOT IN TABLE
1113                   *
1114                           PULX                    SKIP RCBGDT
1115 +  046B 3F                SWI
1116 +  046C 06                FCB 6
1117                           PULX                    GET RCBADR
1118 +  046D 3F                SWI
1119 +  046E 06                FCB 6
1120                   *
1121    046F 0D                SEC                     SET RETURN FLAG
1122                   *
1123    0470 86 05             LDA A #5                ERROR CODE
1124    0472 A4 05             AND A RCBSTA,X
1125    0474 A7 05             STA A RCBSTA,X
1126                   *
1127    0476 39                RTS
1128                   *
1129                   * FOUND ENTRY
1130                   *
1131                   PDSRCB PULX            GET POINTER TO EQT
1132 +  0477 3F                SWI
1133 +  0478 06                FCB 6
1134    0479 EE 00             LDX 0,X        GET ADDRESS OF EQT
1135                           TXAB           SAVE IN A,B
1136 +  047B 3F                SWI
1137 +  047C 02                FCB 2
1138                           PULX           SKIP RCBGDT
1139 +  047D 3F                SWI
1140 +  047E 06                FCB 6
1141                           PULX           GET RCBADR
1142 +  047F 3F                SWI
1143 +  0480 06                FCB 6
1144                           PSHX           SAVE ON STACK
1145 +  0481 3F                SWI
1146 +  0482 05                FCB 5
1147                   *
1148    0483 A7 00             STA A RCBEQT,X         SAVE EQT ADR
1149    0485 E7 01             STA B RCBEQT+1,X
1150                   *
1151    0487 6D 06             TST RCBDTT,X   INPUT OR OUTPUT?
1152    0489 2A 04             BPL *+6        INPUT
1153                   *
1154    048B CB 02             ADD B #2       OUTPUT,POINT TO OUTPUT DRIVER
1155    048D 89 00             ADC A #00
1156                   *
1157                           TABX
1158 +  048F 3F                SWI
1159 +  0490 03                FCB 3
1160    0491 EE 00             LDX 0,X        GET DRIVER ADDRESS
1161                           TXAB           SAVE IN A,B
1162 +  0493 3F                SWI
1163 +  0494 02                FCB 2
1164                           PULX           GET RCBADR
1165 +  0495 3F                SWI
1166 +  0496 06                FCB 6
1167    0497 0C                CLC            SET RETURN FLAG
1168                   *
1169    0498 39                RTS
1171                   * NULL IE BIT BUCKET
1172                   *
1173                   NULL    XABX           DO NOTHING
1174 +  0499 3F                SWI
1175 +  049A 04                FCB 4
1176    049B 39                RTS
1178                   * INLIN: INPUT A LINE FROM THE CONSOLE
1179                   *
1180                   * A,B=RCBADR
1181                   *
1182                   * OBEYS TTYSET PARAMATERS
1183                   *
1184                   INLIN   TABX
1185 +  049C 3F                SWI
1186 +  049D 03                FCB 3
1187                   *
1188                   INLIN1 PSHX                    SAVE RCBADR
1189 +  049E 3F                SWI
1190 +  049F 05                FCB 5
1191    04A0 EE 07             LDX RCBDBA,X           GET BUFFER ADDRESS
1192                   *
1193    04A2 86 2E             LDA A #'.              ISSUE PROMPT
1194    04A4 BD 057D R         JSR OUTCON
1195                   *
1196    04A7 BD 0568 R INLIN2  JSR INCON              GET A CHARACTER
1197    04AA 81 0A             CMP A #$0A             LF?
1198    04AC 27 F9             BEQ INLIN2             YES
1199                   *
1200    04AE 7D 0040           TST DX                 HALF-DUPLEX?
1201    04B1 26 03             BNE *+5                NO
1202                   *
1203    04B3 BD 057D R         JSR OUTCON             ECHO
1204    04B6 91 3A             CMP A DL               DELETE LINE?
1205    04B8 26 04             BNE INLIN3             NO
1206                   *
1207                           PULX                   YES,GET RCBADR
1208 +  04BA 3F                SWI
1209 +  04BB 06                FCB 6
1210    04BC 20 E0             BRA INLIN1
1211                   *
1212    04BE 91 39     INLIN3  CMP A BS               BACK SPACE
1213    04C0 26 16             BNE INLIN4             NO
1214                   *
1215                           TXAB                   YES,SAVE BUFFER PTR
1216 +  04C2 3F                SWI
1217 +  04C3 02                FCB 2
1218                           PULX                   GET RCBADR
1219 +  04C4 3F                SWI
1220 +  04C5 06                FCB 6
1221                   *
1222    04C6 A1 07             CMP A RCBDBA,X         AT START OF BUFFER?
1223    04C8 26 04             BNE *+6                NO
1224                   *
1225    04CA E1 08             CMP B RCBDBA+1,X
1226    04CC 27 04             BEQ INLIN5             YES
1227                   *
1228    04CE C0 01             SUB B #1               BACK UP PTR
```

```
1229   04D0 82 00          SBC A #00
1230                *
1231                INLIN5 PSHX                SAVE RCBADR
1232 +  04D2 3F            SWI
1233 +  04D3 05            FCB 5
1234                       TABX                X POINT TO BUFFER
1235 +  04D4 3F            SWI
1236 +  04D5 03            FCB 3
1237   04D6 20 CF          BRA INLIN2
1238                *
1239   04D8 A7 00     INLIN4 STA A 0,X           STORE CHAR
1240   04DA 08              INX                 BUMP BUFFER POINTER
1241   04DB 81 0D           CMP A #$0D          CR?
1242   04DD 26 C8           BNE INLIN2          NO
1243                *
1244   04DF 86 0A     INLIN6 LDA A #$0A          LF
1245   04E1 BD 057D R       JSR OUTCON
1246                *
1247                       PULX                GET RCBADR
1248 +  04E4 3F            SWI
1249 +  04E5 06            FCB 6
1250   04E6 39              RTS
1252                * OTLIN:  OUTPUT A LINE TO CONSOLE
1253                *
1254                * A,B=RCBADR
1255                *
1256                OTLIN  TABX                X:=RCBADR
1257 +  04E7 3F            SWI
1258 +  04E8 03            FCB 3
1259                       PSHX                SAVE
1260 +  04E9 3F            SWI
1261 +  04EA 05            FCB 5
1262                *
1263   04EB EE 07           LDX RCBDBA,X        GET BUFFER ADDRESS
1264   04ED D6 3D           LDA B WD            GET CHARS/LINE
1265                *
1266   04EF A6 00     OTLIN1 LDA A 0,X           GET A CHAR
1267   04F1 81 04           CMP A #$04          END-STRING?
1268   04F3 27 70           BEQ OTLIN8          YES, DONE
1269                *
1270   04F5 BD 057D R       JSR OUTCON          SEND IT
1271   04F8 08              INX
1272   04F9 5A              DEC B               FULL LINE?
1273   04FA 26 18           BNE OTLINC          NO
1274                *
1275   04FC 86 0D           LDA A #$0D          CR
1276   04FE BD 057D R       JSR OUTCON
1277   0501 86 0A           LDA A #$0A          LF
1278   0503 BD 057D R       JSR OUTCON
1279   0506 86 00           LDA A #00
1280   0508 D6 3E           LDA B NL            NULLS?
1281   050A 27 06           BEQ OTLINB          NO
1282                *
1283   050C BD 057D R OTLINA JSR OUTCON          NULL
1284   050F 5A              DEC B
1285   0510 26 FA           BNE OTLINA
1286                *
1287   0512 D6 3D     OTLINB LDA B WD            GET LINE WIDTH
1288                *
1289   0514 09        OTLINC DEX                 POINT TO LAST CHAR
1290   0515 A6 00           LDA A 0,X
```

```
1291   0517 08              INX                 RESET POINTER
1292                *
1293   0518 81 0D           CMP A #$0D          CR?
1294   051A 26 D3           BNE OTLIN1          NO
1295                *
1296   051C 86 0A           LDA A #$0A          LF
1297   051E BD 057D R       JSR OUTCON
1298                *
1299   0521 86 00           LDA A #00           NULL
1300   0523 D6 3E           LDA B NL            NULLS?
1301   0525 27 06           BEQ OTLIN3          NO
1302                *
1303   0527 BD 057D R OTLIN2 JSR OUTCON
1304   052A 5A              DEC B
1305   052B 26 FA           BNE OTLIN2
1306                *
1307   052D 7D 003B   OTLIN3 TST DP              PAGING ON?
1308   0530 27 21           BEQ OTLIN7          NO
1309                *
1310   0532 7A 003C         DEC DPCNT           PAGE?
1311   0535 26 1C           BNE OTLIN7          NO
1312                *
1313   0537 96 3B           LDA A DP            RESET DPCNT
1314   0539 97 3C           STA A DPCNT
1315                *
1316   053B 7D 0042         TST PS              PAUSE?
1317   053E 26 07           BNE OTLIN5          NO
1318                *
1319   0540 BD 0568 R OTLIN4 JSR INCON          GET A CHAR
1320   0543 91 43           CMP A ES            ESCAPE?
1321   0545 26 F9           BNE OTLIN4          NO
1322                *
1323   0547 D6 41     OTLIN5 LDA B EJ            GET EJECT COUNT
1324   0549 27 08           BEQ OTLIN7          NO EJECTS
1325                *
1326   054B 86 0A     OTLIN6 LDA A #$0A          LF
1327   054D BD 057D R       JSR OUTCON
1328   0550 5A              DEC B               DONE?
1329   0551 26 F8           BNE OTLIN6          NO
1330                *
1331                OTLIN7 PULX                GET RCBADR
1332 +  0553 3F            SWI
1333 +  0554 06            FCB 6
1334                       PSHX                RESTACK
1335 +  0555 3F            SWI
1336 +  0556 05            FCB 5
1337   0557 EE 00           LDX RCBEQT,X        GET EQT ADDRESS
1338   0559 EE 04           LDX 4,X             GET PHYSICAL ADDRESS
1339                *
1340   055B A6 00           LDA A 0,X           GET STATUS
1341   055D 47              ASR A
1342   055E 24 05           BCC OTLIN8          NO BREAK
1343                *
1344   0560 A6 01           LDA A 1,X           READ DATA
1345   0562 BD 0568 R       JSR INCON           WAIT FOR ANY INPUT
1346                *
1347                OTLIN8 PULX                GET RCBADR
1348 +  0565 3F            SWI
1349 +  0566 06            FCB 6
1350                *
1351   0567 39              RTS
```

```
                        * INCON:  GET A CHAR FROM THE CONSOLE
                        * X:=BUFFER ADDRESS
                        * A=CHAR PARITY STRIPPED
                        *
                        INCON  PSHX           SAVE BUFFER PTR
0568 3F                  SWI
0569 05                  FCB 5
056A 30                 TSX
056B EE 04              LDX 4,X       GET RCBADR
056D EE 00              LDX RCBEQT,X  GET EQT ADDRESS
056F EE 04              LDX 4,X       GET PHYSICAL ADDR
                        *
0571 A6 00              INCON1 LDA A 0,X      GET STAT
0573 47                 ASR A
0574 24 FB              BCC INCON1    NOT READY
                        *
0576 A6 01              LDA A 1,X     GET CHAR
0578 84 7F              AND A #$7F    STRIP PARITY
                        *
                        PULX          GET BUFFER ADDR
057A 3F                  SWI
057B 06                  FCB 6
                        *
057C 39                 RTS
                        * OUTPUT A CHAR TO CONSOLE
                        * X:=BUFFER ADDRESS; CHAR IN A
                        *
                        OUTCON PSHX           SAVE BUFFER ADDRESS
057D 3F                  SWI
057E 05                  FCB 5
057F 30                 TSX
0580 EE 04              LDX 4,X       GET RCBADR
0582 EE 00              LDX RCBEQT,X  GET EQT ADDR
0584 EE 04              LDX 4,X       GET PHYSICAL ADDR
0586 37                 PSH B
                        *
0587 E6 00              OUTCO1 LDA B 0,X      GET STATUS
0589 57                 ASR B
058A 57                 ASR B
058B 24 FA              BCC OUTCO1    NOT READY
                        *
058D A7 01              STA A 1,X     SEND CHAR
058F 33                 PUL B
                        PULX          GET BUFFER ADDR
0590 3F                  SWI
0591 06                  FCB 6
                        *
0592 39                 RTS
                        * READ A LINE FROM PAPER TAPE READER
                        *
                        * A,B=RCBADR
                        *
                        INRDR  TABX           X:=RCBADR
0593 3F                  SWI
0594 03                  FCB 3
                        PSHX          STACK
0595 3F                  SWI
0596 05                  FCB 5
0597 EE 07              LDX RCBDBA,X  GET BUFFER ADDRESS
                        *
0599 86 11              LDA A #$11    X-ON
059B BD 05ED R          JSR OUTPCH
                        *
                        *
059E BD 05B8 R          INRD1  JSR RDRIN      GET A CHAR
05A1 81 0A              CMP A #$0A    LF?
05A3 27 F9              BEQ INRD1     YES
                        *
05A5 81 00              CMP A #$00    NULL?
05A7 27 F5              BEQ INRD1     YES
                        *
05A9 A7 00              STA A 0,X
05AB 08                 INX
05AC 81 0D              CMP A #$0D    CR?
05AE 26 EE              BNE INRD1     NO
                        *
05B0 86 13              INRD2  LDA A #$13     X-OFF
05B2 BD 05ED R          JSR OUTPCH
                        *
                        PULX          GET RCBADR
05B5 3F                  SWI
05B6 06                  FCB 6
                        *
05B7 39                 RTS
                        * READ A CHAR FROM PAPER TAPE READER
                        *
                        * X:=BUFFER PTR RETURN IN A
                        *
                        RDRIN  PSHX           SAVE BUFFER ADDR
05B8 3F                  SWI
05B9 05                  FCB 5
05BA 30                 TSX
05BB EE 04              LDX 4,X       GET RCBADR
05BD EE 00              LDX RCBEQT,X  GET EQT ADDR
05BF EE 04              LDX 4,X       GET PHYSICAL ADDRESS
                        *
05C1 A6 00              RDRIN1 LDA A 0,X
05C3 47                 ASR A
05C4 24 FB              BCC RDRIN1    NOT READY
                        *
05C6 A6 01              LDA A 1,X     GET BYTE
                        PULX          GET BUFFER PTR
05C8 3F                  SWI
05C9 06                  FCB 6
                        *
05CA 39                 RTS
                        * PUNCH A LINE OF PAPER TAPE
                        *  WITH NULLS BETWEEN LINES
                        * A,B=RCBADR
                        *
                        OTPCH  TABX           X:=RCBADR
05CB 3F                  SWI
05CC 03                  FCB 3
                        PSHX          STACK RCBADR
05CD 3F                  SWI
05CE 05                  FCB 5
05CF EE 07              LDX RCBDBA,X  GET BUFFER ADDRESS
                        *
05D1 A6 00              OTPCH1 LDA A 0,X      GET A CHAR
05D3 08                 INX
05D4 BD 05ED R          JSR OUTPCH
05D7 81 0D              CMP A #$0D    CR?
```

```
1479   05D9 26 F6            BNE OTPCH1    NO
1480                       *
1481   05DB 86 0A            LDA A #$0A    LF
1482   05DD BD 05ED R        JSR OUTPCH
1483                       *
1484   05E0 C6 04            LDA B #4
1485                       *
1486   05E2 86 00     OTPCH3 LDA A #00     NULLS
1487   05E4 BD 05ED R        JSR OUTPCH
1488   05E7 5A               DEC B
1489   05E8 26 F8            BNE OTPCH3    NOT DONE
1490                       *
1491                         PULX          RESTORE RCBADR
1492 +  05EA 3F               SWI
1493 +  05EB 06               FCB 6
1494   05EC 39               RTS
1496                       * PUNCH A CHARACTER ON PUNCH
1497                       * X=BUFFER POINTER
1498                       *
1499                       OUTPCH PSHX          SAVE BUFFER PTR
1500 +  05ED 3F               SWI
1501 +  05EE 05               FCB 5
1502   05EF 30               TSX
1503   05F0 EE 04            LDX 4,X       GET RCBADR
1504   05F2 EE 00            LDX RCBEQT,X  GET EQT ADR
1505   05F4 EE 04            LDX 4,X       GET PHYSICAL ADDR
1506   05F6 37               PSH B
1507                       *
1508   05F7 E6 00     OUTPC1 LDA B 0,X     GET STATUS
1509   05F9 57               ASR B
1510   05FA 57               ASR B
1511   05FB 24 FA            BCC OUTPC1    NOT READY
1512                       *
1513   05FD A7 01            STA A 1,X     SEND BYTE
1514   05FF 33               PUL B
1515                         PULX          GET BUFFER PTR
1516 +  0600 3F               SWI
1517 +  0601 06               FCB 6
1518                       *
1519   0602 39               RTS
1521                       * OUTPUT A LINE TO PRINTER
1522                       * A,B=RCBADR
1523                       *
1524                       OTLPT  TABX          X:=RCBADR
1525 +  0603 3F               SWI
1526 +  0604 03               FCB 3
1527                         PSHX          STACK
1528 +  0605 3F               SWI
1529 +  0606 05               FCB 5
1530   0607 EE 07            LDX RCBDBA,X          GET BUFFER ADDRESS
1531   0609 D6 46            LDA B LWD             GET CHARS/LINE
1532                       *
1533   060B A6 00     OTLPT1 LDA A 0,X     GET A CHAR
1534   060D 08               INX
1535   060E BD 0648 R        JSR OUTLPT    PRINT
1536   0611 5A               DEC B                 FULL LINE?
1537   0612 26 0C            BNE OTLPT2            NO
1538                       *
1539   0614 86 0D            LDA A #$0D            CR
1540   0616 BD 0648 R        JSR OUTLPT
1541   0619 86 0A            LDA A #$0A            LF
1542   061B BD 0648 R        JSR OUTLPT
1543   061E D6 46            LDA B LWD             GET CHARS/LINE
1544                       *
1545   0620 09        OTLPT2 DEX
1546   0621 A6 00            LDA A 0,X
1547   0623 08               INX
1548   0624 81 0C            CMP A #$0C            FF ?
1549   0626 26 06            BNE OTLP3             NO
1550                       *
1551   0628 37               PSH B
1552   0629 D6 44            LDA B LDP             RESET LINES/PAGE
1553   062B D7 45            STA B LDPCNT
1554   062D 33               PUL B
1555                       *
1556   062E 81 0D     OTLP3 CMP A #$0D     CR?
1557   0630 26 D9            BNE OTLPT1    NO
1558                       *
1559   0632 86 0A            LDA A #$0A    LF
1560   0634 BD 0648 R        JSR OUTLPT
1561   0637 7A 0045          DEC LDPCNT    PAGE?
1562   063A 26 09            BNE OTLPT4    NO
1563                       *
1564   063C D6 44            LDA B LDP     INIT LDPCNT
1565   063E D7 45            STA B LDPCNT
1566                       *
1567   0640 86 0C            LDA A #$0C
1568   0642 BD 0648 R        JSR OUTLPT
1569                       *
1570                       OTLPT4 PULX          GET RCBADR
1571 +  0645 3F               SWI
1572 +  0646 06               FCB 6
1573                       *
1574   0647 39               RTS
1576                       * PRINT A CHAR ON LINE PRINTER
1577                       *
1578                       * X=BUFFER PTR
1579                       *
1580                       OUTLPT PSHX          SAVE BUFFER PTR
1581 +  0648 3F               SWI
1582 +  0649 05               FCB 5
1583   064A 30               TSX
1584   064B EE 04            LDX 4,X       GET RCBADR
1585   064D EE 00            LDX RCBEQT,X  GET EQT ADDR
1586   064F EE 04            LDX 4,X       GET PHYSICAL ADDR
1587   0651 37               PSH B
1588                       *
1589   0652 E6 00            LDA B 0,X     CLEAR ACK
1590   0654 A7 00            STA A 0,X     OUTPUT
1591   0656 E6 01            LDA B 1,X     ACK?
1592   0658 2A FC            BPL *-2       NO
1593                       *
1594   065A 33               PUL B         YES
1595                         PULX
1596 +  065B 3F               SWI
1597 +  065C 06               FCB 6
1598   065D 39               RTS
1600                       * READ A SINGLE SECTOR
1601                       * A,B=RCBADR
1602                       *
1603   065E 7E 0000 X        EXT .RDSEC
1604                       *
```

```
                        * WRITE A SINGLE SECTOR
                        * A,B=RCBADR
                        *
0661 7E 0000 X          WTSEC   EXT  WTSEC
                        *
                        * READ A TAPE BLOCK
                        * A,B=RCBADR
                        *
0664 7E 0000 X          MTIN    EXT  MTIN
                        *
                        * WRITE A TAPE BLOCK
                        * A,B=RCBADR
                        *
0667 7E 0000 X          MTOT    EXT  MTOT
                        *
                        *
066A 7E 0000 X          @OPEN   EXT @OPEN     OPEN SEQUENTIAL FILE
                        *
066D 7E 0000 X          @CLOSE  EXT @CLOSE    CLOSE SEQUENTIAL FILE
                        *
0670 7E 0000 X          @REWD   EXT @REWD     REWIND SEQUENTIAL FILE
                        *
0673 7E 0000 X          @OPEND  EXT @OPEND    OPEN DIRECTORY
                        *
0676 7E 0000 X          @READ   EXT @READ     READ A SEQUENTIAL FILE
                        *
0679 7E 0000 X          @WRITE  EXT @WRITE    WRITE A SEQUENTIAL FILE
                        *
067C 7E 0000 X          @GETDR  EXT @GETDR    GET A DIRECTORY ENTRY
                        *
067F 7E 0000 X          @PUTDR  EXT @PUTDR    WRITE A DIRECTORY ENTRY
                        *
0682 7E 0000 X          @DELET  EXT @DELET    DELETE A FILE
                        *
0685 7E 0000 X          @CHAIN  EXT @CHAIN    CHAIN A PROGRAM FILE
                        *
0688 7E 0000 X          @PRTERR EXT @PRTERR   PRINT AN ERROR MESSAGE
                        *
068B 7E 0000 X          @WRMST  EXT @WRMST    WARM START ENTRY
                        *
068E 7E 0000 X          @USR6   EXT @USR6     USER DEFINED 6
                        *
0691 7E 0000 X          @USR7   EXT @USR7     USER DEFINED 7
                        *
0694 7E 0000 X          @USR8   EXT @USR8     USER DEFINED 8
                        *
0697 7E 0000 X          @USR9   EXT @USR9     USER DEFINED 9
                        *
069A 7E 0000 X          @USR10  EXT @USR10    USER DEFINED 10
                        *
069D 7E 0000 X          @LOADB  EXT @LOADB    LOAD A BINARY FILE
                        *
06A0 7E 0000 X          @LOADR  EXT @LOADR    LOAD A RELOCATABLE FILE
                        *
06A3 7E 0000 X          @NXTOK  EXT @NXTOK    PARSE A COMMAND LINE
                        *
06A6 7E 0000 X          @GTCMD  EXT @GTCMD    GET A COMMAND LINE (CALLS @NXTOK)
                        *
06A9 7E 0000 X          @PRTMSG EXT @PRTMSG   PRINT AN ERROR MESSAGE
                        *
06AC 7E 0000 X          @INTDK  EXT @INTDK    INITIALIZE DISK SYSTEM
                        *
06AF 7E 0000 X          @USR1   EXT @USR1     USER DEFINED 1
                        *
06B2 7E 0000 X          @USR2   EXT @USR2     USER DEFINED 2
                        *
06B5 7E 0000 X          @USR3   EXT @USR3     USER DEFINED 3
                        *
06B8 7E 0000 X          @USR4   EXT @USR4     USER DEFINED 4
                        *
06BB 7E 0000 X          @USR5   EXT @USR5     USER DEFINED 5
                        *
06BE 7E 0000 X          @FMTFCB EXT @FMTFCB
                        *
                        END
```

```
.MTIN  0664 RX   ADDAB  00CC R    FMTS9  0331 R    OTLIN8 0565 R    TESTZA 00DB R
.MTOT  0667 RX   ADDABX 2219 M    FMTSA  0263 R    OTLINA 050C R    TO     000B
.RDSEC 065E RX   ADDAX  2232 M    FMTSB  0269 R    OTLINB 0512 R    TTYIO  03E4 R
.WTSEC 0661 RX   ADDBX  224B M    FMTSC  0278 R    OTLINC 0514 R    TXAB   2183 M
@ADABX 00C7 R    ADDXAB 2200 M    FRETAB 002B      OTLP3  062E R    UA     0004
@ADDAX 00DE R    ADDZ   00E1 R    FROM   0009      OTLPT  0603 R    UB     0003
@ADDBX 00E5 R    BASEQU 2A2A M    GETDR  23EC M    OTLPT1 060B R    UC     0002
@ADXAB 00C0 R    BIOS   0000 RN   GTCMD  24F0 M    OTLPT2 0620 R    URH    0007
@CHAIN 0685 RX   BMEM   0033      INCON  0568 R    OTLPT4 0645 R    URL    0008
@CLDST 0000 RX   BS     0039      INCON1 0571 R    OTPCH  05CB R    UXH    0005
@CLOSE 066D RX   BUSY   0080      INDEX  24BC M    OTPCH1 05D1 R    UXL    0006
@CMPC  01D5 R    CDONE  01FF R    INITDK 253E M    OTPCH3 05E2 R    VALUE  0027
@CMWC  0206 R    CHAIN  243A M    INLIN  049C R    OUTC01 0587 R    WD     003D
@DELET 0682 RX   CLASS  0026      INLIN1 049E R    OUTCON 057D R    WRITE  23D2 M
@DIV16 015F R    CLOSE  2369 M    INLIN2 04A7 R    OUTLPT 0648 R    XABX   21B5 M
@FMTFC 06BE RX   CMEM   0037      INLIN3 04BE R    OUTPC1 05F7 R
@FMTS  0254 R    CMPO   01D8 R    INLIN4 04D8 R    OUTPCH 05ED R
@GETDR 067C RX   CMP1   01F6 R    INLIN5 04D2 R    PARM1  0009
@GTCMD 06A6 RX   CMP2   01FC R    INLIN6 04DF R    PARM2  000B
@INDEX 0111 R    CMPC   231B M    INRD1  059E R    PDSRCA 0449 R
@INTDK 06AC RX   CMWO   0209 R    INRD2  05B0 R    PDSRCB 0477 R
@IOHDR 0429 R    CMW1   0224 R    INRDR  0593 R    PDSRCH 0440 R
@LOADB 069D RX   CMWC   2572 M    IOHDR  2335 M    PDTAB  03F0 RN
@LOADR 06A0 RX   CONSOL 03C0 R    IOHDRA 0439 R    PRTERR 2454 M
@MOVC  01B6 R    CUCHAR 0023      JMPTZ  0122 R    PRTMSG 250A M
@MOVS  0236 R    DELETE 2420 M    LDP    0044      PS     0042
@MUL16 0124 R    DESCRA 0020      LDPCNT 0045      PSHALC 0050 R
@MUL8  0119 R    DESCRC 0022      LOADB  246E M    PSHALL 2151 M
@NXTOK 06A3 RX   DISK   03D2 R    LPTR   03D8 R    PSHALS 0045 R
@OPEN  066A RX   DIV151 0172 R    LWD    0046      PSHX   21CE M
@OPEND 0673 RX   DIV153 017D R    MOVC   2301 M    PSHXA  0098 R
@PRTER 0688 RX   DIV16  2524 M    MOVC1  01B9 R    PTPCH  03CC R
@PRTMS 06A9 RX   DIV163 0187 R    MOVC2  01CC R    PTRDR  03C6 R
@PSHAL 003D R    DIV165 0194 R    MOVC3  01D4 R    PULAC  005C R
@PSHX  0093 R    DIV167 0195 R    MOVS   24A2 M    PULALS 0066 R
@PULAL 0059 R    DL     003A      MOVS1  0237 R    PULLAL 216A M
@PULX  00AA R    DP     003B      MOVS2  024B R    PULX   21E7 M
@PUTDR 067F RX   DPCNT  003C      MOVS3  0253 R    PULXA  00B5 R
@READ  0676 RX   DSPTAB 0340      MPY8   014B R    PUTDR  2406 M
@REWD  0670 RX   DX     0040      MPY8L  0153 R    RC     0025
@SBABX 00F1 R    EJ     0041      MPY8S  0157 R    RCBDBA 0007
@SBXAB 00EA R    EMEM   0035      MTAPE  03DE R    RCBDEF 258C M
@SUB   0101 R    EQTAB  03C0 RN   MUL16  22E7 M    RCBDTT 0006
@SUBAX 00FE R    ES     0043      MUL16L 012E R    RCBEQT 0000
@SUBBX 010C R    FCBCHN 0029      MUL16S 0134 R    RCBGDT 0002
@TABX  007E R    FCBDEF 2650 M    MUL8   22CD M    RCBSTA 0005
@TXAB  0074 R    FIBDEF 2940 M    NL     003E      RDRIN  05B8 R
@USR1  06AF RX   FMTFCB 2488 M    NTBUSY 007F      RDRIN1 05C1 R
@USR10 069A RX   FMTS   2558 M    NULL   0499 R    READ   23B8 M
@USR2  06B2 RX   FMTS0  025D R    NULLIO 03EA R    REWIND 2384 M
@USR3  06B5 RX   FMTS1  027E R    NXTOK  24D6 M    STAB   0079 R
@USR4  06B8 RX   FMTS10 0350 R    OPEN   234F M    START  0014 R
@USR5  06BB RX   FMTS2  0284 R    OPEND  239E M    STAUXH 00D2 R
@USR6  068E RX   FMTS3  0292 R    OTLIN  04E7 R    SUBABX 227F M
@USR7  0691 RX   FMTS4  02B2 R    OTLIN1 04EF R    SUBAX  2299 M
@USR8  0694 RX   FMTS5  02D7 R    OTLIN2 0527 R    SUBBX  22B3 M
@USR9  0697 RX   FMTS5A 02D9 R    OTLIN3 052D R    SUBXAB 2265 M
@WRITE 0679 RX   FMTS5B 0300 R    OTLIN4 0540 R    SWIHDR 0003 R
@WRMST 068B RX   FMTS6  030B R    OTLIN5 0547 R    TABX   219C M
@XABX  0088 R    FMTS7  031C R    OTLIN6 054B R    TB     003F
                 FMTS8  0328 R    OTLIN7 0553 R    TEST7  00D7 R
```

```
0001   0000 0000     N          NAM CLI          COMMAND LINE INTERPRETER
0002                             *
0003                             *       COPYRIGHT 1978 BY HEMENWAY ASSOCIATES INC
0004                             *       BOSTON MASS.  02111
0005                             *       ALL RIGHTS RESERVED
0006                             *
0007                             * COLD START ENTRY POINT
0008                             *
0009   0000 7E 02A3 R @CLDST JMP COLDST
0010                             *
0011                             * WARM START ENTRY POINT
0012                             *
0013   0003 7E 0320 R @WRMST JMP WARMST
0014                             *
0015   0006 7E 0000 X           EXT EQTAB
0016   0009 7E 0000 X           EXT PDTAB
0017                             *
0018   000C 0000     N          ENT @CLDST
0019   000C 0003     N          ENT @WRMST
0020   000C 0471     N          ENT @NXTOK
0021   000C 0397     N          ENT @PRTMSG
0022   000C 03D4     N          ENT @PRTERR
0023   000C 03FE     N          ENT @GTCMD
0024   000C 0966     N          ENT @LOADB
0025   000C 0E48     N          ENT @CHAIN
0026   000C 0F5E     N          ENT @DELETE
0027   000C 108D     N          ENT @FMTFCB
0028   000C 0564     N          ENT CVHB
0029   000C 05BB     N          ENT CVDB
0030   000C 0EC5     N          ENT SEMPTY
0031   000C 0EFE     N          ENT SFILE
0032   000C 000C     N          ENT SYSFCB
0033                             *
0034                             *
0035                             * BASE PAGE EQUATES
0036                             *
0037                                    BASEQU
0038 +  000C 0020                 DESCRA EQU $20  DESCRIPTOR ADDRESS(2)
0039 +  000C 0022                 DESCRC EQU $22  DESCRIPTOR COUNT
0040 +  000C 0023                 CUCHAR EQU $23  CURRENT CHAR (2)
0041 +  000C 0025                 RC     EQU $25  TOKEN RETURN CODE
0042 +  000C 0026                 CLASS  EQU $26  TOKEN CLASS
0043 +  000C 0027                 VALUE  EQU $27  BIN VALUE/TRANSFER ADDRESS (2)
0044 +  000C 0029                 FCBCHN EQU $29  TOP OF FCB CHAIN (2)
0045 +  000C 002B                 FRETAB EQU $2B  DISK FREE SPACE POINTER (8)
0046 +  000C 0033                 BMEM   EQU $33  START OF TRANSIENT AREA(2)
0047 +  000C 0035                 EMEM   EQU $35  END OF TRANSIENT AREA (2)
0048 +  000C 0037                 CMEM   EQU $37  NEXT AVAIL TRANSIENT AREA (2)
0049 +  000C 0039                 BS     EQU $39  BACKSPACE CHAR
0050 +  000C 003A                 DL     EQU $3A  DELETE LINE CHAR
0051 +  000C 003B                 DP     EQU $3B  DEPTH; LINES/PAGE
0052 +  000C 003C                 DPCNT  EQU $3C  DEPTH TEMP
0053 +  000C 003D                 WD     EQU $3D  WIDTH; CHARS/LINE
0054 +  000C 003E                 NL     EQU $3E  NULL COUNT
0055 +  000C 003F                 TB     EQU $3F  TAB CHAR
0056 +  000C 0040                 DX     EQU $40  DUPLEX; FF=H, 00=F
0057 +  000C 0041                 EJ     EQU $41  EJECT COUNT
0058 +  000C 0042                 PS     EQU $42  PAUSE; 00=YES
0059 +  000C 0043                 ES     EQU $43  ESCAPE CHAR
0060 +  000C 0044                 LDP    EQU $44  DEPTH LINES/PAGE
```

```
0061 +  000C 0045      LDPCNT EQU $45  DEPTH TEMP
0062 +  000C 0046      LWD    EQU $46  WIDTH CHARS/LINE
0063                   *
0064                   * BLOCK ADDRESSING DEFINITIONS
0065                   *
0066                         RCBDEF
0067 +  000C 0000      RCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0068 +  000C 0002      RCBGDT EQU 2          GENERIC DEVICE TYPE
0069 +  000C 0005      RCBSTA EQU 5          STATUS
0070 +  000C 0006      RCBDTT EQU 6          DATA TRANSFER TYPE
0071 +  000C 0007      RCBDBA EQU 7          DATA BUFFER ADDRESS
0072                         FCBDEF
0073 +  000C 0000      FCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0074 +  000C 0002      FCBGDT EQU 2          GENERIC DEVICE TYPE
0075 +  000C 0005      FCBSTA EQU 5          STATUS
0076 +  000C 0006      FCBDTT EQU 6          DATA TRANSFER TYPE
0077 +  000C 0007      FCBDBA EQU 7          DATA BUFFER ADDRESS
0078 +  000C 0009      FCBDRV EQU 9          DRIVE NUMBER
0079 +  000C 000A      FCBTRK EQU 10         TRACK NUMBER
0080 +  000C 000B      FCBSCT EQU 11         SECTOR NUMBER
0081 +  000C 000C      FCBFWD EQU 12         FWD LINK TRACK/SECTOR
0082 +  000C 000E      FCBBAK EQU 14         BACK LINK TRACK/SECTOR
0083 +  000C 0010      FCBNAM EQU 16         FILE NAME (8.3+EOT=13))
0084 +  000C 001D      FCBTYP EQU 29         FILE TYPE
0085 +  000C 001E      FCBACS EQU 30         FILE ACCESS CODE
0086 +  000C 001F      FCBFTS EQU 31         FIRST TRACK/SECTOR
0087 +  000C 0021      FCBLTS EQU 33         LAST TRACK/SECTOR
0088 +  000C 0023      FCBNMS EQU 35         NUMBER OF SECTORS
0089 +  000C 0025      FCBNFB EQU 37         NEXT FCB IN ACTIVE CHAIN
0090 +  000C 0027      FCBIND EQU 39         INDEX INTO DATA BUFFER
0091 +  000C 0029      FCBSCF EQU 41         SPACE COMPRESSION FLAG
0092                         FIBDEF
0093 +  000C 0000      FIBNAM EQU 0          FILE NAME (8.3 + EOT=13)
0094 +  000C 000D      FIBTYP EQU 13         FILE TYPE
0095 +  000C 000E      FIBACS EQU 14         FILE ACCESS CODE
0096 +  000C 000F      FIBFTS EQU 15         FIRST TRACK/SECTOR
0097 +  000C 0011      FIBLTS EQU 17         LAST TRACK/SECTOR
0098 +  000C 0013      FIBNMS EQU 19         NUMBER OF SECTORS
0099                   *
0100                   * REGISTER OFFSETS ON 'BIOS' CALLS
0101                   *
0102    000C 0003      UB     EQU 3
0103    000C 0004      UA     EQU 4
0104    000C 0005      UXH    EQU 5
0105    000C 0006      UXL    EQU 6
0106                   *
0107                   * DISK ATTRIBUTES
0108                   *
0109    000C 0080      SECSIZ  EQU 128      128 BYTES/SECTOR
0110                   *
0111                   * SYSTEM FILE-CONTROL BLOCK
0112                   *
0113    000C 0002      SYSFCB RMB 2
0114    000E 44              FCC 'DSK'
0115    0011 0002            RMB 2
0116    0013 0036    R       FDB BUFFER
0117    0015 0021            RMB 33
0118                   *
0119    0036 0080      BUFFER RMB SECSIZ    DATA SPACE
0120                   *
0121                   * STACK SPACE FOR SYSTEM
0122                   *
0123    00B6 0100            RMB 256
0124    01B6 0001      STACK  RMB 1
0126                   * COMMAND TABLE
0127                   *
0128    01B7 44        CMDTAB FCC 'DIR'     DIRECTORY
0129    01BA 0BE5    R       FDB DIRCMD
0130                   *
0131    01BC 50              FCC 'PIP'     PERIPHERAL INTERCHANGE PROGRAM
0132    01BF 0620    R       FDB PIPCMD
0133                   *
0134    01C1 52              FCC 'REN'     RENAME
0135    01C4 09F1    R       FDB RENCMD
0136                   *
0137    01C6 44              FCC 'DEL'     DELETE
0138    01C9 061B    R       FDB DELCMD
0139                   *
0140    01CB 53              FCC 'SAV'     SAVE
0141    01CE 07B4    R       FDB SAVCMD
0142                   *
0143    01D0 41              FCC 'ASS'     ASSIGN
0144    01D3 062F    R       FDB ASNCMD
0145                   *
0146    01D5 4C              FCC 'LOA'     LOAD
0147    01D8 0956    R       FDB LODCMD
0148                   *
0149    01DA 4A              FCC 'JUM'     JUMP
0150    01DD 0607    R       FDB JMPCMD
0151                   *
0152    01DF 49              FCC 'INI'     INITIALIZE
0153    01E2 0AF9    R       FDB INICMD
0154                   *
0155    01E4 45              FCC 'EXI'     EXIT
0156    01E7 E0E3            FDB $E0E3     MONITOR
0157                   *
0158    01E9 53              FCC 'SEC'     SECURITY
0159    01EC 0625    R       FDB SECCMD
0160                   *
0161    01EE 53              FCC 'SET'     SETCON
0162    01F1 062A    R       FDB SETCMD
0163                   *
0164    01F3 53              FCC 'STA'     STATUS
0165    01F6 0634    R       FDB STACMD
0166                   *
0167    01F8 42              FCC 'BOO'     BOOTSTRAP
0168    01FB 0639    R       FDB BOOTCD
0169                   *
0170    01FD 4C              FCC 'LIN'     LINK
0171    0200 063E    R       FDB LNKCMD
0172                   *
0173    0202 53              FCC 'SUB'     SUBMIT
0174    0205 06C6    R       FDB SUBCMD
0175                   *
0176    0207 00              FCB 00        END OF TABLE
0178                   * CHARACTER TABLE
0179                   *
0180    0208 00        CHRTAB FCB  $00  SPACE
0181    0209 00               FCB  $00  !
0182    020A 00               FCB  $00  "
0183    020B 00               FCB  $00  #
0184    020C 04               FCB  $04  $
```

```
0185   020D 00              FCB  $00  %
0186   020E 00              FCB  $00  &
0187   020F 00              FCB  $00  '
0188   0210 00              FCB  $00  (
0189   0211 00              FCB  $00  )
0190   0212 81              FCB  $81  *
0191   0213 04              FCB  $04  +
0192   0214 04              FCB  $04  ,
0193   0215 04              FCB  $04  -
0194   0216 04              FCB  $04  .
0195   0217 04              FCB  $04  /
0196   0218 42              FCB  $42  0
0197   0219 42              FCB  $42  1
0198   021A 42              FCB  $42  2
0199   021B 42              FCB  $42  3
0200   021C 42              FCB  $42  4
0201   021D 42              FCB  $42  5
0202   021E 42              FCB  $42  6
0203   021F 42              FCB  $42  7
0204   0220 42              FCB  $42  8
0205   0221 42              FCB  $42  9
0206   0222 04              FCB  $04  :
0207   0223 04              FCB  $04  ;
0208   0224 00              FCB  $00  <
0209   0225 04              FCB  $04  =
0210   0226 00              FCB  $00  >
0211   0227 81              FCB  $81  ?
0212   0228 00              FCB  $00  @
0213   0229 82              FCB  $82  A
0214   022A 82              FCB  $82  B
0215   022B 82              FCB  $82  C
0216   022C 82              FCB  $82  D
0217   022D 82              FCB  $82  E
0218   022E 82              FCB  $82  F
0219   022F 80              FCB  $80  G
0220   0230 80              FCB  $80  H
0221   0231 80              FCB  $80  I
0222   0232 80              FCB  $80  J
0223   0233 80              FCB  $80  K
0224   0234 80              FCB  $80  L
0225   0235 80              FCB  $80  M
0226   0236 80              FCB  $80  N
0227   0237 80              FCB  $80  O
0228   0238 80              FCB  $80  P
0229   0239 80              FCB  $80  Q
0230   023A 80              FCB  $80  R
0231   023B 80              FCB  $80  S
0232   023C 80              FCB  $80  T
0233   023D 80              FCB  $80  U
0234   023E 80              FCB  $80  V
0235   023F 80              FCB  $80  W
0236   0240 80              FCB  $80  X
0237   0241 80              FCB  $80  Y
0238   0242 80              FCB  $80  Z
0239   0243 04              FCB  $04  [
0240   0244 00              FCB  $00  \
0241   0245 04              FCB  $04  ]
0242   0246 00              FCB  $00  CAROT
0243   0247 00              FCB  $00  UNDERLINE
0244   0248 0050    CONBUF  RMB  80       CONSOLE BUFFER
0245   0298 000B    CONRCB  RMB  11       CONSOLE RCB
0246                *
0247   02A3 8E 01B6 R COLDST LDS  #STACK    INITIALIZE STACK POINTER
0248   02A6 7F 0707 R        CLR  SUBFLG   INIT. 'SUBMIT' FLAG
0249   02A9 FE 0007 R        LDX  EQTAB+1  POINT TO EQTAB
0250   02AC C6 04            LDA B #4      POINT TO CONSOLE ENTRY
0251                         ADDBX
0252 +  02AE 3F              SWI
0253 +  02AF 0A              FCB  10
0254   02B0 EE 00            LDX  0,X      GET ACIA ADDRESS
0255   02B2 86 03            LDA A #3
0256   02B4 A7 00            STA A 0,X     RESET ACIA
0257   02B6 86 15            LDA A #$15
0258   02B8 A7 00            STA A 0,X     INIT ACIA
0259                *
0260   02BA FE 0007 R        LDX  EQTAB+1  POINT TO EQTAB
0261   02BD C6 28            LDA B #40     POINT TO TTY ENTRY
0262                         ADDBX
0263 +  02BF 3F              SWI
0264 +  02C0 0A              FCB  10
0265   02C1 EE 00            LDX  0,X      GET ACIA ADDRESS
0266   02C3 86 03            LDA A #3
0267   02C5 A7 00            STA A 0,X     RESET ACIA
0268   02C7 86 01            LDA A #1
0269   02C9 A7 00            STA A 0,X     INIT ACIA
0270                *
0271   02CB FE 0007 R        LDX  EQTAB+1  POINT TO EQTAB
0272   02CE C6 1C            LDA B #28     POINT TO LPT ENTRY
0273                         ADDBX
0274 +  02D0 3F              SWI
0275 +  02D1 0A              FCB  10
0276   02D2 EE 00            LDX  0,X      GET PIA ADDRESS
0277   02D4 86 FF            LDA A #$FF
0278   02D6 A7 00            STA A 0,X     DDRB:=OUTPUT
0279   02D8 86 2C            LDA A #$2C
0280   02DA A7 01            STA A 1,X     INIT PIA
0281                *
0282                         INITDK        INITIALIZE DISK SYSTEM
0283 +  02DC 3F              SWI
0284 +  02DD 33              FCB  51
0285                *
0286                * INIT CONSOLE RCB
0287                *
0288   02DE CE 0298 R        LDX  #CONRCB
0289   02E1 86 43            LDA A #'C
0290   02E3 A7 02            STA A RCBGDT,X
0291   02E5 86 4F            LDA A #'O
0292   02E7 A7 03            STA A RCBGDT+1,X
0293   02E9 86 4E            LDA A #'N
0294   02EB A7 04            STA A RCBGDT+2,X
0295                *
0296   02ED CE 02FE R        LDX  #CRLF    ISSUE A CR,LF
0297                         PRTMSG
0298 +  02F0 3F              SWI
0299 +  02F1 31              FCB  49
0300   02F2 CE 0300 R        LDX  #BANNER  PRINT HEADLINE
0301                         PRTMSG
0302 +  02F5 3F              SWI
0303 +  02F6 31              FCB  49
0304   02F7 CE 0000          LDX  #0       INIT. ACTIVE FCB CHAIN
0305   02FA DF 29            STX  FCBCHN
0306   02FC 20 32            BRA  WARM3    NOW PROCESS COMMANDS
```

```
0307                    *
0308   02FE 0A0D        CRLF    FDB $0A0D       LF,CR
0309   0300 0A          BANNER  FCB $0A
0310   0301 48                  FCC 'HEMENWAY ASSOCIATES CP/68-1.0'
0311   031E 0A0D                FDB $0A0D
0312                    *
0313   0320 8E 01B6 R   WARMST  LDS #STACK      REINIT STACK POINTER
0314   0323 7F 0707 R           CLR SUBFLG      INIT. 'SUBMIT' FLAG
0315   0326 DE 29       WARM1   LDX FCBCHN      ANY ACTIVE FCBS?
0316   0328 27 06               BEQ WARM3       NO
0317                    *
0318                    * CLOSE ALL ACTIVE (OPEN) FCBS
0319                    *
0320                    WARM2   CLOSE
0321 +  032A 3F                  SWI
0322 +  032B 15                  FCB 21
0323   032C EE 25               LDX FCBNFB,X    POINT TO NEXT FCB
0324   032E 26 FA               BNE WARM2
0325                    *
0326   0330 CE 0000     WARM3   LDX #0
0327   0333 DF 2B               STX FRETAB      INIT. FREE-SPACE TABLE
0328   0335 DF 2D               STX FRETAB+2    4 DRIVES
0329   0337 DF 2F               STX FRETAB+4
0330   0339 DF 31               STX FRETAB+6
0331                    *
0332                    CLIO    GTCMD           GET COMMAND
0333 +  033B 3F                  SWI
0334 +  033C 30                  FCB 48
0335   033D D6 25               LDA B RC        CHECK RC
0336   033F C1 01               CMP B #1        UNAMBIG. NAME?
0337   0341 27 0B               BEQ CLI2        YES
0338                    *
0339   0343 C1 03               CMP B #3        NUMBER?
0340   0345 27 2C               BEQ TFILE       IF SO, FILENAME
0341                    *
0342   0347 CE 03AA R   CLI1    LDX #FORMAT     NO
0343                            PRTMSG          PRINT ERROR
0344 +  034A 3F                  SWI
0345 +  034B 31                  FCB 49
0346   034C 20 ED               BRA CLIO
0347                    *
0348                    * LOOK UP COMMAND
0349                    *
0350   034E DE 20       CLI2    LDX DESCRA
0351                            PSHX            STACK PARM
0352 +  0350 3F                  SWI
0353 +  0351 05                  FCB 5
0354   0352 CE 01B7 R           LDX #CMDTAB
0355                    *
0356                    CMDSRA  PSHX            STACK PARM
0357 +  0355 3F                  SWI
0358 +  0356 05                  FCB 5
0359   0357 C6 03               LDA B #3        COUNT OF THREE
0360                            CMPC
0361 +  0359 3F                  SWI
0362 +  035A 12                  FCB 18
0363   035B 27 2C               BEQ CMDSRB      MATCH
0364                    *
0365                    * NO MATCH
0366                    *
0367                            PULX            GET CMDTAB PTR
0368 +  035D 3F                  SWI
0369 +  035E 06                  FCB 6
0370                            ADDBX           POINT TO NEXT ENTRY
0371 +  035F 3F                  SWI
0372 +  0360 0A                  FCB 10
0373   0361 08                  INX                     "
0374   0362 08                  INX                     "
0375                            TXAB            SAVE PTR TO CMDTAB
0376 +  0363 3F                  SWI
0377 +  0364 02                  FCB 2
0378                            PULX            GET DESCRA
0379 +  0365 3F                  SWI
0380 +  0366 06                  FCB 6
0381   0367 DE 20               LDX DESCRA
0382                            PSHX            STACK PARM
0383 +  0369 3F                  SWI
0384 +  036A 05                  FCB 5
0385                            TABX            RESTORE CMDTAB PTR
0386 +  036B 3F                  SWI
0387 +  036C 03                  FCB 3
0388   036D 6D 00               TST 0,X         END OF TABLE?
0389   036F 26 E4               BNE CMDSRA      NO
0390                    *
0391                    * NOT IN TABLE
0392                    *
0393                            PULX            SKIP DESCRA PARM
0394 +  0371 3F                  SWI
0395 +  0372 06                  FCB 6
0396                    *
0397                    * PROCESS AS A TRANSIENT COMMAND FILE-NAME
0398                    *
0399   0373 DE 20       TFILE   LDX DESCRA
0400   0375 DF 23               STX CUCHAR      BACK UP ONE TOKEN ON CLI
0401   0377 BD 0956 R           JSR LODCMD      TRY TO LOAD TRANSIENT
0402   037A CE 08AC R           LDX #SAVFCB
0403   037D 6D 05               TST FCBSTA,X    GOOD LOAD?
0404   037F 26 AF               BNE WARM3       IF NOT, QUIT
0405                    *
0406   0381 DE 27               LDX VALUE       HAVE TRANSFER ADDRESS?
0407   0383 27 AB               BEQ WARM3       IF NOT, QUIT
0408                    *
0409   0385 AD 00               JSR 0,X         IF SO, CALL IT
0410   0387 20 A7               BRA WARM3       DONE
0411                    *
0412                    *
0413                    * FOUND ENTRY
0414                    *
0415                    CMDSRB  PULX            GET PTR TO ROUTINE
0416 +  0389 3F                  SWI
0417 +  038A 06                  FCB 6
0418   038B EE 00               LDX 0,X         GET ADDRESS OF ROUTINE
0419                            TXAB            TRANSFER TO A,B
0420 +  038D 3F                  SWI
0421 +  038E 02                  FCB 2
0422                            PULX            SKIP DESCRA PARM
0423 +  038F 3F                  SWI
0424 +  0390 06                  FCB 6
0425                            TABX            RESTORE ROUTINE ADDRESS
0426 +  0391 3F                  SWI
0427 +  0392 03                  FCB 3
0428   0393 AD 00               JSR 0,X         CALL ROUTINE
```

```
0429                    *
0430   0395 20 99               BRA WARM3
0432                    * PRINT ERROR MESSAGE; X=A(MESSAGE)
0433                    *
0434   0397 30          @PRTMSG TSX
0435   0398 A6 05               LDA A UXH,X    SAVE ADDRESS IN A,B
0436   039A E6 06               LDA B UXL,X
0437   039C CE 0298 R           LDX #CONRCB    POINT TO CONSOLE RCB
0438   039F A7 07               STA A RCBDBA,X
0439   03A1 E7 08               STA B RCBDBA+1,X
0440   03A3 86 80               LDA A #$80
0441   03A5 A7 06               STA A RCBDTT,X        DIRECTION:=OUTPUT
0442                    *
0443                            IOHDR          OUTPUT MESSAGE
0444 +  03A7 3F                 SWI
0445 +  03A8 13                 FCB 19
0446                    *
0447   03A9 39                  RTS
0448                    *
0449                    * MESSAGES
0450                    *
0451   03AA 46          FORMAT FCC 'FORMAT ERROR'
0452   03B6 0D                  FCB $0D
0453                    *
0454   03B7 4E          NUMBER FCC 'NUMBER ERROR'
0455   03C3 0D                  FCB $0D
0456                    *
0457   03C4 20          DEVERR FCC ' '
0458   03C5 0003        DEVNAM RMB 3
0459   03C8 20                  FCC ' ERROR: '
0460   03D0 0002        DERNUM RMB 2
0461   03D2 0A0D                FDB $0A0D
0462                    *
0463                    *
0464                    * PRINT AN ERROR MESSAGE ON RCB OR FCB STATUS
0465                    * INDEX POINTS TO CONTROL BLOCK
0466                    *
0467                    * PRINT DEVICE NAME AND STATUS
0468                    *
0469   03D4 30          @PRTERR TSX
0470   03D5 EE 05               LDX UXH,X              POINT TO CONTROL BLOCK
0471   03D7 A6 05               LDA A FCBSTA,X         GET DEVICE STATUS
0472   03D9 27 22               BEQ PRTER2             IF ZERO, QUIT
0473                    *
0474   03DB BD 0E39 R           JSR OUTHL              CONVERT LEFT DIGIT TO ASCII
0475   03DE B7 03D0 R           STA A DERNUM
0476   03E1 A6 05               LDA A FCBSTA,X
0477   03E3 BD 0E3D R           JSR OUTHR              CONVERT RIGHT DIGIT
0478   03E6 B7 03D1 R           STA A DERNUM+1
0479   03E9 A6 02               LDA A RCBGDT,X         GET DEVICE NAME
0480   03EB B7 03C5 R           STA A DEVNAM
0481   03EE A6 03               LDA A RCBGDT+1,X
0482   03F0 B7 03C6 R           STA A DEVNAM+1
0483   03F3 A6 04               LDA A RCBGDT+2,X
0484   03F5 B7 03C7 R           STA A DEVNAM+2
0485   03F8 CE 03C4 R           LDX #DEVERR
0486                            PRTMSG                 PRINT ERROR LINE
0487 +  03FB 3F                 SWI
0488 +  03FC 31                 FCB 49
0489                    *
0490   03FD 39          PRTER2 RTS
```

```
0492                    * GET A COMMAND FROM THE CONSOLE
0493                    * PARSE THE FIRST TOKEN FROM THE COMMAND LINE
0494                    *
0495   03FE CE 0298 R @GTCMD LDX #CONRCB    POINT TO RCB
0496   0401 6F 06               CLR RCBDTT,X  SET FOR INPUT
0497   0403 86 43               LDA A #'C     'CON' SOLE DEVICE
0498   0405 A7 02               STA A RCBGDT,X
0499   0407 86 4F               LDA A #'O
0500   0409 A7 03               STA A RCBGDT+1,X
0501   040B 86 4E               LDA A #'N
0502   040D A7 04               STA A RCBGDT+2,X
0503                            TXAB
0504 +  040F 3F                 SWI
0505 +  0410 02                 FCB 2
0506   0411 CE 0248 R           LDX #CONBUF    POINT TO BUFFER
0507                            XABX
0508 +  0414 3F                 SWI
0509 +  0415 04                 FCB 4
0510   0416 A7 07               STA A RCBDBA,X        INIT DATA BUFFER ADDRESS
0511   0418 E7 08               STA B RCBDBA+1,X
0512   041A 7D 0707 R           TST SUBFLG     IN 'SUBMIT'?
0513   041D 27 4B               BEQ GTCD2      NO
0514                    *
0515                    * GET COMMAND FROM 'SUBMIT' FILE
0516                    *
0517   041F CE 0248 R           LDX #CONBUF
0518   0422 FF 0708 R           STX SUBTMP     SAVE BUFFER POINTER
0519   0425 CE 070A R SUBMT1 LDX #SUBFCB
0520                            READ           READ A CHARACTER
0521 +  0428 3F                 SWI
0522 +  0429 18                 FCB 24
0523   042A E6 05               LDA B FCBSTA,X
0524   042C C1 08               CMP B #8       END-FILE?
0525   042E 27 0B               BEQ SUBEOF     YES
0526                    *
0527   0430 5D                  TST B          ERROR STATUS?
0528   0431 27 12               BEQ SUBMT2     NO
0529                    *
0530   0433 BD 06E7 R           JSR SUBERR     ISSUE ERROR MESSAGE AND CLOSE FILE
0531   0436 7F 0707 R           CLR SUBFLG     RESET 'SUBMIT' FLAG
0532   0439 20 C3               BRA @GTCMD     GET CONSOLE COMMAND
0533                    *
0534   043B CE 070A R SUBEOF LDX #SUBFCB
0535                            CLOSE          CLOSE FILE
0536 +  043E 3F                 SWI
0537 +  043F 15                 FCB 21
0538   0440 7F 0707 R           CLR SUBFLG     RESET 'SUBMIT' FLAG
0539   0443 20 B9               BRA @GTCMD     GET CONSOLE COMMAND
0540                    *
0541   0445 FE 0708 R SUBMT2 LDX SUBTMP     POINT TO BUFFER
0542   0448 A7 00               STA A 0,X      STORE CHARACTER
0543   044A 08                  INX
0544   044B FF 0708 R           STX SUBTMP
0545   044E 81 0D               CMP A #$0D     C.R.?
0546   0450 26 09               BNE SUBMT3     NO
0547                    *
0548   0452 CE 0248 R           LDX #CONBUF
0549                            PRTMSG         ECHO COMMAND LINE
0550 +  0455 3F                 SWI
0551 +  0456 31                 FCB 49
0552   0457 DF 23               STX CUCHAR     INIT. POINTER FOR NXTOK
```

```
0553   0459 20 16              BRA @NXTOK
0554                   *
0555   045B 81 26      SUBMT3  CMP A #'&      'DIVERSION'?
0556   045D 26 05              BNE SUBMT4     NO
0557                   *
0558   045F CE 0298 R          LDX #CONRCB
0559   0462 20 06              BRA GTCD2      GET A LINE FROM CONSOLE
0560                   *
0561   0464 81 04      SUBMT4  CMP A #$04     CNTL-D? (EOF IN TEXT FILE)
0562   0466 27 D3              BEQ SUBEOF
0563                   *
0564   0468 20 BB              BRA SUBMT1     GET NEW CHARACTER
0565                   *
0566                   GTCD2   IOHDR          GET A LINE FROM THE CONSOLE
0567 +  046A 3F                 SWI
0568 +  046B 13                 FCB 19
0569   046C CE 0248 R          LDX #CONBUF
0570   046F DF 23              STX CUCHAR     INIT PARMS FOR NXTOK
0571                   *
0572                   * FALL INTO 'NXTOK' ROUTINE
0573                   *
0575                   * NEXT TOKEN ROUTINE *
0576                   * SCANS A LINE OF SOURCE CODE AND RETURNS
0577                   * THE NEXT TOKEN CLASS AND RC
0578                   * THE ADDRESS OF THE TOKEN IS RETURNED IN
0579                   * DESCRA AND THE # OF BYTES IN THE TOKEN IS
0580                   * RETURNED IN DESCRC.
0581                   * THE RC AND CLASS ARE:
0582                   *
0583                   * TYPE:    RC [B]    CLASS [A]
0584                   *
0585                   * NAME      01          02  SUBSTRINGS
0586                   * WCARD     02          02
0587                   * NUMBER    03          02
0588                   *
0589                   * DELIMS    (ASCII)     04  DELIMITERS
0590                   *
0591                   * CR        0D          0D  EOL
0592                   *
0593                   * ERROR     00          00  ERRORS
0594                   *
0595                   *
0596   0471 7F 0022    @NXTOK  CLR   DESCRC
0597   0474 7C 0022            INC   DESCRC     DESCRC:=1
0598   0477 DE 23      NXTO    LDX   CUCHAR     POINT TO CURRENT CHAR
0599   0479 DF 20              STX   DESCRA     INIT DESCRA
0600   047B A6 00              LDA A 0,X        GET CHAR
0601   047D 08                 INX
0602   047E DF 23              STX   CUCHAR     POINT TO NEXT CHAR
0603   0480 81 20              CMP A #$20       LESS THAN 20 HEX?
0604   0482 27 F3              BEQ   NXTO       BLANK, SKIP OVER
0605   0484 22 07              BHI   NXT1       >20
0606                   *
0607   0486 81 0D              CMP A #$0D       CR?
0608   0488 26 3A              BNE   NXTER      NO, UNRECOG.  CHAR
0609   048A 16                 TAB              YES,  SET RC
0610   048B 20 32              BRA   NXT7
0611                   *
0612   048D 81 5F      NXT1    CMP A #$5F       >5F?
0613   048F 23 02              BLS   NXT3       NO
0614   0491 20 31              BRA   NXTER      YES, UNRECOG.  CHAR
0615                   *
0616   0493 BD 0550 R  NXT3    JSR   GCHRTB     GET BYTE FROM CHARTAB
0617                   *
0618   0496 85 80      NXT4    BIT A #$80       NAME?
0619   0498 27 0D              BEQ   NXT5       NO
0620                   *
0621   049A C6 01              LDA B #1
0622   049C 85 01              BIT A #$01       WILDCARD CHAR?
0623   049E 27 02              BEQ *+4          NO
0624                   *
0625   04A0 C6 02              LDA B #2         YES
0626                   *
0627   04A2 BD 04F1 R          JSR   NSCAN      YES SCAN NAME STRING
0628   04A5 20 18              BRA   NXT7
0629                   *
0630   04A7 85 40      NXT5    BIT A #$40       DECIMAL?
0631   04A9 27 05              BEQ   NXT6       NO
0632                   *
0633   04AB BD 04CB R          JSR   DSCAN      YES, SCAN DECIMAL STRING
0634   04AE 20 0F              BRA   NXT7
0635                   *
0636   04B0 85 04      NXT6    BIT A #$04       DELIMITERS?
0637   04B2 27 10              BEQ   NXTER      NO, UNRECOG.  CHAR
0638                   *
0639   04B4 DE 20              LDX   DESCRA     GET CHAR
0640   04B6 E6 00              LDA B 0,X
0641   04B8 C1 24              CMP B #$24       $? (HEX)
0642   04BA 26 03              BNE   NXT7       NO, GET RC AND RTN
0643                   *
0644   04BC BD 0513 R          JSR   HSCAN      YES, SCAN HEX STRING
0645                   *
0646   04BF 97 26      NXT7    STA A CLASS
0647   04C1 D7 25              STA B RC
0648   04C3 39                 RTS
0649                   *
0650   04C4 7F 0026    NXTER   CLR CLASS        TROUBLE, SET RC, CLASS=00
0651   04C7 7F 0025            CLR RC
0652   04CA 39                 RTS
0654                   * DSCAN SCAN DECIMAL STRING STOP AT
0655                   * FIRST NON-DECIMAL CHAR
0656                   *
0657   04CB DE 23      DSCAN   LDX   CUCHAR     POINT TO NEXT CHAR
0658   04CD A6 00              LDA A 0,X        GET CHAR
0659   04CF 7C 0022            INC   DESCRC     BUMP COUNT
0660   04D2 08                 INX
0661   04D3 DF 23              STX   CUCHAR     POINT TO NEXT CHAR
0662   04D5 BD 0550 R          JSR   GCHRTB     GET BYTE IN CHARTAB
0663   04D8 85 40              BIT A #$40       DECIMAL?
0664   04DA 26 EF              BNE   DSCAN      YES CONTINUE SCAN
0665                   *
0666   04DC D6 22              LDA B DESCRC     CHECK LENGTH OF TOKEN
0667   04DE 5A                 DEC B
0668   04DF D7 22              STA B DESCRC     WENT ONE CHAR.  TOO FAR
0669   04E1 C1 06              CMP B #6
0670   04E3 2D 03              BLT   DSCAN2     LENGTH OK?
0671                   *
0672   04E5 7E 0546 R          JMP   NUMERR     NO,  ERROR
0673                   *
0674   04E8 BD 05BB R  DSCAN2  JSR   CVDB       CONVERT TO BINARY
0675   04EB DF 27              STX   VALUE      SAVE VALUE
0676   04ED C6 03              LDA B #3
```

```
0677   04EF 20 4D          BRA   ENDSCN    FINISH UP
0678                 *
0679                 *
0680                 * NSCAN SCAN NAME STRING STOP AT
0681                 * FIRST NON-ALPHANUMERIC CHAR
0682                 *
0683   04F1 DE 23    NSCAN LDX   CUCHAR    POINT TO NEXT CHAR
0684   04F3 A6 00          LDA A 0,X       GET CHAR
0685   04F5 7C 0022        INC   DESCRC    BUMP COUNT
0686   04F8 08             INX
0687   04F9 DF 23          STX   CUCHAR    POINT TO NEXT CHAR
0688   04FB BD 0550 R      JSR   GCHRTB    GET BYTE IN CHARTAB
0689   04FE 85 01          BIT A #$01      WILDCARD?
0690   0500 27 02          BEQ   *+4       NO
0691                 *
0692   0502 C6 02          LDA B #2        YES
0693                 *
0694   0504 85 80          BIT A #$80      ALPHA?
0695   0506 26 E9          BNE   NSCAN     YES CONTINUE SCAN
0696                 *
0697   0508 85 40          BIT A #$40      NUMERIC?
0698   050A 26 E5          BNE   NSCAN     YES CONTINUE SCAN
0699                 *
0700   050C 96 22          LDA A DESCRC
0701   050E 4A             DEC A
0702   050F 97 22          STA A DESCRC    WENT ONE CHAR.  TOO FAR
0703   0511 20 2B          BRA ENDSCN      FINISH UP
0704                 *
0705                 *
0706                 * HSCAN SCAN HEX STRING STOP AT
0707                 * FIRST NON-HEX CHAR
0708                 *
0709   0513 7F 0022  HSCAN CLR   DESCRC    DESCRC:=0
0710   0516 DE 23          LDX   CUCHAR    POINT TO NEXT CHAR
0711   0518 DF 20          STX   DESCRA    INIT DESCRA
0712   051A DE 23    HSCAN1 LDX   CUCHAR   POINT TO NEXT CHAR
0713   051C A6 00          LDA A 0,X       GET CHAR
0714   051E 7C 0022        INC   DESCRC    BUMP COUNT
0715   0521 08             INX
0716   0522 DF 23          STX   CUCHAR    POINT TO NEXT CHAR
0717   0524 BD 0550 R      JSR   GCHRTB    GET BYTE IN CHRTAB
0718   0527 85 02          BIT A #$02      HEX?
0719   0529 26 EF          BNE   HSCAN1    YES CONTINUE SCAN
0720                 *
0721   052B D6 22          LDA B DESCRC    CHECK TOKEN LENGTH
0722   052D 5A             DEC B
0723   052E D7 22          STA B DESCRC    WENT ONE CHAR.  TOO FAR
0724   0530 C1 05          CMP B #5
0725   0532 2D 03          BLT   HSCAN2    LENGTH OK?
0726                 *
0727   0534 7E 0546 R      JMP   NUMERR    NO, ERROR
0728                 *
0729   0537 BD 0564 R HSCAN2 JSR   CVHB      CONVERT HEX TO BINARY
0730   053A DF 27          STX   VALUE     SAVE BINARY VALUE
0731   053C C6 03          LDA B #3
0732                 *
0733   053E DE 23    ENDSCN LDX   CUCHAR
0734   0540 09             DEX
0735   0541 DF 23          STX   CUCHAR    CUCHAR:= CORRECT VALUE
0736   0543 86 02          LDA A #2        LOAD CLASS RC
0737   0545 39             RTS             ALL DONE
0738                 *
0739                 *
0740   0546 CE 03B7 R NUMERR LDX #NUMBER     PRINT 'NUMBER ERROR' MESSAGE
0741                       PRTMSG
0742 +  0549 3F            SWI
0743 +  054A 31            FCB 49
0744   054B 31             INS             CLEAN STACK
0745   054C 31             INS
0746   054D 7E 04C4 R      JMP NXTER       RETURN ERROR TOKEN
0748                 * GET BYTE IN CHRTAB INDEXED BY VALUE OF
0749                 * CHAR IN REG A
0750                 *
0751   0550 81 20    GCHRTB CMP A #$20      VALID CHAR ?
0752   0552 25 0C          BCS   GCHRTR    NO < 20
0753   0554 81 5F          CMP A #$5F      VALID CHAR ?
0754   0556 22 08          BHI   GCHRTR    NO, > 5F
0755                 *
0756   0558 CE 01E8 R      LDX   #CHRTAB-$20
0757                       ADDAX           ADD IN CHARACTER OFFSET
0758 +  055B 3F            SWI
0759 +  055C 09            FCB 9
0760   055D A6 00          LDA A 0,X       GET BYTE
0761   055F 39             RTS
0762                 *
0763   0560 4F       GCHRTR CLR A
0764   0561 39             RTS
0765                 *
0767                 * CVHB CONVERT HEX TO BINARY
0768                 *
0769                 * ON ENTRY DESCRA = ADDRESS OF STRING
0770                 *          DESCRC = # OF BYTES IN STRING
0771                 * ON RETURN [X]=VALUE
0772                 *
0773   0562 0002     HVAL  RMB   2           TEMP STORAGE
0774                 *
0775   0564 DE 20    CVHB  LDX   DESCRA      GET ADDRESS OF STRING
0776   0566 7F 0562 R      CLR   HVAL
0777   0569 7F 0563 R      CLR   HVAL+1
0778   056C D6 22          LDA B DESCRC    GET COUNT
0779   056E 09             DEX             DECREMENT PTR TO STRING
0780   056F 08       CVHB1 INX             POINT TO RIGHT MOST
0781   0570 5A             DEC B           BYTE OF THE
0782   0571 26 FC          BNE   CVHB1     STRING
0783                 *
0784   0573 D6 22          LDA B DESCRC    GET COUNT
0785   0575 BD 05AB R      JSR   CVHBS     CONVERT
0786   0578 B7 0563 R      STA A HVAL+1    SAVE
0787   057B 5A             DEC B           DECREMENT COUNT
0788   057C 27 29          BEQ   CVHBD     (1 HEX DIGIT)
0789   057E 09             DEX             POINT TO NEXT LEFT BYTE
0790   057F BD 05AB R      JSR   CVHBS     CONVERT
0791   0582 48             ASL A           SHIFT TO LEFT NIBBLE
0792   0583 48             ASL A
0793   0584 48             ASL A
0794   0585 48             ASL A
0795   0586 BA 0563 R      ORA A HVAL+1    CONVERT TO BYTE
0796   0589 B7 0563 R      STA A HVAL+1    SAVE
0797   058C 5A             DEC B           DECREMENT COUNT
0798   058D 27 18          BEQ   CVHBD     (2 HEX DIGITS)
0799   058F 09             DEX             POINT TO NEXT LEFT BYTE
0800   0590 BD 05AB R      JSR   CVHBS     CONVERT
```

```
0801   0593 B7 0562 R        STA A HVAL      SAVE
0802   0596 5A               DEC B           DECREMENT COUNT
0803   0597 27 0E            BEQ   CVHBD     (3 HEX DIGITS)
0804                  *
0805   0599 09               DEX             POINT TO NEXT LEFT BYTE
0806   059A BD 05AB R        JSR   CVHBS     CONVERT
0807   059D 48               ASL A           SHIFT TO LEFT NIBBLE
0808   059E 48               ASL A
0809   059F 48               ASL A
0810   05A0 48               ASL A
0811   05A1 BA 0562 R        ORA A HVAL      CONVERT TO BYTE
0812   05A4 B7 0562 R        STA A HVAL      SAVE
0813   05A7 FE 0562 R CVHBD  LDX   HVAL      GET FINAL VALUE
0814   05AA 39               RTS             RETURN
0815                  *
0816                  * ROUTINE TO CONVERT ASCII TO BINARY
0817                  *
0818   05AB A6 00     CVHBS  LDA A 0,X       GET BYTE
0819   05AD 80 30            SUB A #$30      CONVERT
0820   05AF 81 09            CMP A #$09      0 - 9 ?
0821   05B1 2F 02            BLE   *+4       YES
0822   05B3 80 07            SUB A #$07      NO, 10 - 15
0823   05B5 39               RTS
0825                  * CVDB: CONVERT DECIMAL TO BINARY
0826                  * ON ENTRY DESCRA = ADDRESS OF DECIMAL STRING
0827                  *          DESCRC = # BYTES IN DECIMAL STRING
0828                  * ON RETURN [X] = VALUE IN BINARY
0829                  *
0830   05B6 0002      DVAL   RMB   2          TEMP STORAGE FOR BINARY
0831   05B8 0001      DCOUNT RMB   1          DIGIT COUNT
0832   05B9 0002      TENVL  RMB   2          POWER OF TEN
0833                  *
0834   05BB 7F 05B6 R CVDB   CLR   DVAL       DVAL:=0
0835   05BE 7F 05B7 R        CLR   DVAL+1
0836   05C1 7F 05B9 R        CLR   TENVL
0837   05C4 7F 05BA R        CLR   TENVL+1
0838   05C7 7C 05BA R        INC   TENVL+1    TENVL:=1
0839   05CA DE 20            LDX   DESCRA     POINT TO STRING
0840   05CC 09               DEX
0841   05CD D6 22            LDA B DESCRC
0842   05CF F7 05B8 R        STA B DCOUNT     INIT DCOUNT
0843                  *
0844   05D2 08        CVDB1  INX              POINT TO
0845   05D3 5A               DEC B           LEAST SIGNIFICANT
0846   05D4 26 FC            BNE   CVDB1     DIGIT
0847                  *
0848                  CVDB2  PSHX             SAVE POINTER
0849 +  05D6 3F               SWI
0850 +  05D7 05               FCB 5
0851   05D8 E6 00            LDA B 0,X       GET DIGIT
0852   05DA C4 0F            AND B #$0F      CONVERT TO BCD
0853   05DC 4F               CLR A           CLEAR ACCUMULATOR
0854   05DD FE 05B9 R        LDX   TENVL     POINT TO POWER OF TEN
0855                         MUL16           (X):=TENVL*DIGIT
0856 +  05E0 3F               SWI
0857 +  05E1 10               FCB 16
0858                         TXAB            (A,B)=(X)
0859 +  05E2 3F               SWI
0860 +  05E3 02               FCB 2
0861   05E4 FB 05B7 R        ADD B DVAL+1     DVAL:=DVAL+TENVL*DIGIT
0862   05E7 B9 05B6 R        ADC A DVAL
0863   05EA B7 05B6 R        STA A DVAL
0864   05ED F7 05B7 R        STA B DVAL+1
0865   05F0 4F               CLR A
0866   05F1 C6 0A            LDA B #$0A       B:=10
0867   05F3 FE 05B9 R        LDX   TENVL      POINT TO POWER OF TEN
0868                         MUL16            (X):=TENVL*10
0869 +  05F6 3F               SWI
0870 +  05F7 10               FCB 16
0871   05F8 FF 05B9 R        STX   TENVL
0872                         PULX             RESTORE POINTER TO STRING
0873 +  05FB 3F               SWI
0874 +  05FC 06               FCB 6
0875   05FD 09               DEX              POINT NEXT LEFT DIGIT
0876   05FE 7A 05B8 R        DEC   DCOUNT     DONE?
0877   0601 26 D3            BNE   CVDB2      NO
0878   0603 FE 05B6 R        LDX   DVAL       GET FINAL VALUE
0879   0606 39               RTS              RETURN
0880                  *
0882                  *
0883                  * PROCESS 'JUMP' COMMAND
0884                  *
0885                  JMPCMD NXTOK            GET ADDRESS
0886 +  0607 3F               SWI
0887 +  0608 2F               FCB 47
0888   0609 D6 25            LDA B RC         CHECK RC
0889   060B C1 03            CMP B #3         VALID NUMBER?
0890   060D 27 06            BEQ JMPC2        YES
0891                  *
0892   060F CE 03AA R        LDX #FORMAT      NO, ERROR
0893                         PRTMSG
0894 +  0612 3F               SWI
0895 +  0613 31               FCB 49
0896   0614 39               RTS              RETURN TO CLI
0897                  *
0898   0615 31        JMPC2  INS              CLEAN STACK
0899   0616 31               INS
0900   0617 DE 27            LDX VALUE        LOAD ADDRESS
0901   0619 6E 00            JMP 0,X          GO THERE
0902                  *
0904                  * PROCESS TRANSIENT COMMANDS
0905                  *
0906                  * 'DELETE'    'PIP'      'SECURITY'    'SETCON'
0907                  * 'ASSIGN'    'STATUS'   'LINK'        'BOOT'
0908                  *
0909   061B CE 0666 R DELCMD LDX #DELLIN
0910   061E 20 21            BRA TRANS
0911                  *
0912   0620 CE 0673 R PIPCMD LDX #PIPLIN
0913   0623 20 1C            BRA TRANS
0914                  *
0915   0625 CE 067D R SECCMD LDX #SECLIN
0916   0628 20 17            BRA TRANS
0917                  *
0918   062A CE 068C R SETCMD LDX #SETLIN
0919   062D 20 12            BRA TRANS
0920                  *
0921   062F CE 0696 R ASNCMD LDX #ASNLIN
0922   0632 20 0D            BRA TRANS
0923                  *
0924   0634 CE 06A3 R STACMD LDX #STALIN
0925   0637 20 08            BRA TRANS
```

```
0926                    *
0927   0639 CE 06B0 R BOOTCD LDX #BOOTLN
0928   063C 20 03            BRA TRANS
0929                    *
0930   063E CE 06BB R LNKCMD LDX #LNKLIN
0931                    *
0932                    *
0933                    TRANS  TXAB
0934 +  0641 3F                SWI
0935 +  0642 02                FCB 2
0936   0643 DE 23             LDX CUCHAR          SAVE OLD CUCHAR
0937                           PSHX
0938 +  0645 3F                SWI
0939 +  0646 05                FCB 5
0940                           TABX
0941 +  0647 3F                SWI
0942 +  0648 03                FCB 3
0943   0649 DF 23             STX CUCHAR          SET UP DUMMY CLI
0944   064B BD 0956 R         JSR LODCMD          ISSUE 'LOAD' COMMAND
0945                           PULX
0946 +  064E 3F                SWI
0947 +  064F 06                FCB 6
0948   0650 DF 23             STX CUCHAR          RESTORE OLD CUCHAR
0949   0652 CE 08AC R         LDX #SAVFCB
0950   0655 6D 05             TST FCBSTA,X        GOOD LOAD?
0951   0657 26 0C             BNE TRANS2          NO, QUIT
0952                    *
0953   0659 DE 27             LDX VALUE           TRANSFER ADDRESS?
0954   065B 27 08             BEQ TRANS2          NO, QUIT
0955                    *
0956                    * FOR 'ASSIGN' TRANSIENT, PASS "PDTAB" ADDRESS IN (A,B)
0957   065D B6 000A R         LDA A PDTAB+1
0958   0660 F6 000B R         LDA B PDTAB+2
0959   0663 6E 00             JMP 0,X             YES, GO THERE
0960                    *
0961   0665 39         TRANS2 RTS
0962                    *
0963   0666 30         DELLIN FCC '0:DELETE.CMD'
0964   0672 0D                FCB $0D
0965                    *
0966   0673 30         PIPLIN FCC '0:PIP.CMD'
0967   067C 0D                FCB $0D
0968                    *
0969   067D 30         SECLIN FCC '0:SECURITY.CMD'
0970   068B 0D                FCB $0D
0971                    *
0972   068C 30         SETLIN FCC '0:SET.CMD'
0973   0695 0D                FCB $0D
0974                    *
0975   0696 30         ASNLIN FCC '0:ASSIGN.CMD'
0976   06A2 0D                FCB $0D
0977                    *
0978   06A3 30         STALIN FCC '0:STATUS.CMD'
0979   06AF 0D                FCB $0D
0980                    *
0981   06B0 30         BOOTLN FCC '0:BOOT.CMD'
0982   06BA 0D                FCB $0D
0983                    *
0984   06BB 30         LNKLIN FCC '0:LINK.CMD'
0985   06C5 0D                FCB $0D
0987                    * PROCESS 'SUBMIT' COMMAND
```

```
0988                    *
0989                    * SYNTAX: SUBMIT [DRIVE:] FILENAME.EXT
0990                    * FILE MUST BE 'TEXT' TYPE
0991                    *
0992   06C6 CE 070A R SUBCMD LDX #SUBFCB
0993   06C9 6F 06             CLR FCBDTT,X  MAKE INPUT FILE
0994   06CB 86 FF             LDA A #$FF
0995   06CD A7 29             STA A FCBSCF,X  BLANK EXPANSION ON
0996                           FMTFCB         FORMAT [DRIVE:] FILENAME.EXT
0997 +  06CF 3F                SWI
0998 +  06D0 2C                FCB 44
0999                           PRTERR         PRINT ERROR MESSAGES
1000 +  06D1 3F                SWI
1001 +  06D2 1E                FCB 30
1002   06D3 6D 05             TST FCBSTA,X  ERROR?
1003   06D5 26 10             BNE SUBERR    YES
1004                    *
1005                           OPEN           OPEN FILE
1006 +  06D7 3F                SWI
1007 +  06D8 14                FCB 20
1008   06D9 6D 05             TST FCBSTA,X  ERROR?
1009   06DB 26 0A             BNE SUBERR
1010                    *
1011   06DD A6 1D             LDA A FCBTYP,X 'TEXT' FILE?
1012   06DF 81 03             CMP A #3
1013   06E1 26 04             BNE SUBERR     IF NOT, ERROR
1014                    *
1015   06E3 7C 0707 R         INC SUBFLG     SET 'SUBMIT' FLAG
1016   06E6 39                RTS            DONE!
1017                    *
1018   06E7 CE 06F4 R SUBERR LDX #SUBLIN   'SUBMIT' ERROR
1019                           PRTMSG
1020 +  06EA 3F                SWI
1021 +  06EB 31                FCB 49
1022   06EC CE 070A R         LDX #SUBFCB
1023                           PRTERR         PRINT ERROR MESSAGE
1024 +  06EF 3F                SWI
1025 +  06F0 1E                FCB 30
1026                           CLOSE          TRY TO CLOSE FILE
1027 +  06F1 3F                SWI
1028 +  06F2 15                FCB 21
1029   06F3 39                RTS
1030                    *
1031   06F4 20         SUBLIN FCC ' SUBMIT FILE ERROR'
1032   0706 0D                FCB $0D
1033                    *
1034   0707 0001       SUBFLG RMB 1
1035   0708 0002       SUBTMP RMB 2
1036                    *
1037   070A 0002       SUBFCB RMB 2
1038   070C 44                FCC 'DSK'
1039   070F 0002              RMB 2
1040   0711 0734    R         FDB SUBBUF
1041   0713 0021              RMB 33
1042                    *
1043   0734 0080       SUBBUF RMB SECSIZ
1045                    * PROCESS 'SAVE' COMMAND
1046                    *
1047                    * STORE MEMORY CONTENTS ON DISK IN BINARY FORMAT
1048                    *
1049                    * SYNTAX: SAVE [DRIVE:] FILENAME.EXT,STARTAD,ENDAD [,TRANSAI
```

```
1050                    *
1051   07B4 CE 08AC R SAVCMD LDX #SAVFCB          POINT TO FCB
1052   07B7 6F 1D           CLR FCBTYP,X         BINARY TYPE
1053   07B9 6F 29           CLR FCBSCF,X         NO COMPRESSION
1054   07BB 6F 1E           CLR FCBACS,X         ACCESS-TYPE=0
1055   07BD 86 FF           LDA A #$FF
1056   07BF A7 06           STA A FCBDTT,X       OUTPUT
1057                        FMTFCB               FORMAT [DRIVE:] FILE.EXT
1058 +  07C1 3F             SWI
1059 +  07C2 2C             FCB 44
1060                        PRTERR               PRINT ERROR MESSAGES
1061 +  07C3 3F             SWI
1062 +  07C4 1E             FCB 30
1063   07C5 6D 05           TST FCBSTA,X         ERROR?
1064   07C7 27 01           BEQ *+3
1065                   *
1066   07C9 39              RTS                  IF SO, RETURN
1067                   *
1068                        NXTOK                GET TOKEN FROM CLI
1069 +  07CA 3F             SWI
1070 +  07CB 2F             FCB 47
1071   07CC 96 26           LDA A CLASS
1072   07CE 81 04           CMP A #4             DELIMITER?
1073   07D0 27 06           BEQ SAV5B            YES
1074                   *
1075   07D2 CE 03AA R SAV5  LDX #FORMAT          NO, FORMAT ERROR
1076                        PRTMSG
1077 +  07D5 3F             SWI
1078 +  07D6 31             FCB 49
1079   07D7 39              RTS
1080                   *
1081                   SAV5B NXTOK               GET TOKEN FROM CLI
1082 +  07D8 3F             SWI
1083 +  07D9 2F             FCB 47
1084   07DA D6 25           LDA B RC             CHECK RC
1085   07DC C1 03           CMP B #3             NUMBER?
1086   07DE 26 F2           BNE SAV5             NO, ERROR
1087                   *
1088   07E0 DE 27           LDX VALUE
1089   07E2 FF 0BDD R       STX SAVEX            SAVE STARTING ADDRESS
1090                        NXTOK                GET TOKEN FROM CLI
1091 +  07E5 3F             SWI
1092 +  07E6 2F             FCB 47
1093   07E7 96 26           LDA A CLASS
1094   07E9 81 04           CMP A #4             DELIMITER?
1095   07EB 26 E5           BNE SAV5             NO, ERROR
1096                   *
1097                        NXTOK                GET TOKEN FROM CLI
1098 +  07ED 3F             SWI
1099 +  07EE 2F             FCB 47
1100   07EF D6 25           LDA B RC             CHECK RC
1101   07F1 C1 03           CMP B #3             NUMBER?
1102   07F3 26 DD           BNE SAV5             NO, ERROR
1103                   *
1104   07F5 DE 27           LDX VALUE
1105   07F7 FF 0BDF R       STX SAVEX1           SAVE ENDING ADDRESS
1106                        NXTOK                GET TOKEN FROM CLI
1107 +  07FA 3F             SWI
1108 +  07FB 2F             FCB 47
1109   07FC 96 26           LDA A CLASS
1110   07FE 81 0D           CMP A #$0D           END OF LINE?
1111   0800 27 2B           BEQ SAV6             YES (NO TRANSFER ADDRESS)
1112                   *
1113   0802 81 04           CMP A #4             DELIMITER?
1114   0804 26 CC           BNE SAV5             NO, ERROR
1115                   *
1116                        NXTOK                GET A TOKEN FROM CLI
1117 +  0806 3F             SWI
1118 +  0807 2F             FCB 47
1119   0808 D6 25           LDA B RC             CHECK RC
1120   080A C1 03           CMP B #3             NUMBER?
1121   080C 26 C4           BNE SAV5             NO, ERROR
1122                   *
1123                   * OUTPUT TRANSFER-ADDRESS RECORD HERE
1124                   *
1125   080E CE 08AC R       LDX #SAVFCB          POINT TO FCB
1126   0811 6C 1D           INC FCBTYP,X         COMMMAND TYPE
1127                        OPEN                 OPEN SAVE FILE
1128 +  0813 3F             SWI
1129 +  0814 14             FCB 20
1130   0815 6D 05           TST FCBSTA,X         CHECK STATUS
1131   0817 27 03           BEQ *+5              GOOD?
1132                   *
1133   0819 7E 0899 R       JMP SAVERR           BAD
1134                   *
1135   081C 86 16           LDA A #$16           HEADER BYTE
1136   081E BD 08A1 R       JSR PUTBYT
1137   0821 96 27           LDA A VALUE          HIGH-BYTE OF ADDRESS
1138   0823 BD 08A1 R       JSR PUTBYT
1139   0826 96 28           LDA A VALUE+1        LOW-BYTE OF ADDRESS
1140   0828 BD 08A1 R       JSR PUTBYT
1141   082B 20 09           BRA SAV7
1142                   *
1143                   *
1144                   *
1145   082D CE 08AC R SAV6  LDX #SAVFCB          POINT TO FCB
1146                        OPEN
1147 +  0830 3F             SWI
1148 +  0831 14             FCB 20
1149   0832 6D 05           TST FCBSTA,X         FSAVE-FILE OPEN GOOD?
1150   0834 26 63           BNE SAVERR           BAD
1151                   *
1152                   * NOW WRITE OUT BINARY RECORDS
1153                   *
1154   0836 B6 0BDF R SAV7  LDA A SAVEX1
1155   0839 F6 0BE0 R       LDA B SAVEX1+1       GET END-ADDRESS
1156   083C F0 0BDE R       SUB B SAVEX+1
1157   083F B2 0BDD R       SBC A SAVEX
1158   0842 27 29           BEQ SAV9             ARE THERE > 256 BYTES LEFT?
1159                   *
1160   0844 C6 FF           LDA B #$FF           WRITE FULL RECORD (256 BYTES)
1161   0846 CE 08AC R       LDX #SAVFCB          POINT TO FCB
1162   0849 86 02           LDA A #$02           HEADER BYTE
1163   084B 8D 54           BSR PUTBYT
1164   084D B6 0BDD R       LDA A SAVEX          HIGH-BYTE OF ADDRESS
1165   0850 8D 4F           BSR PUTBYT
1166   0852 B6 0BDE R       LDA A SAVEX+1        LOW-BYTE OF ADDRESS
1167   0855 8D 4A           BSR PUTBYT
1168   0857 17              TBA                  256 BYTES IN COUNT
1169   0858 8D 47           BSR PUTBYT
1170   085A FE 0BDD R SAV8  LDX SAVEX            POINT TO MEMORY ADDRESS
1171   085D A6 00           LDA A 0,X            GET BYTE
```

```
1172   085F 08                INX
1173   0860 FF 0BDD R         STX SAVEX
1174   0863 CE 08AC R         LDX #SAVFCB          POINT TO FCB
1175   0866 8D 39             BSR PUTBYT
1176   0868 5A                DEC B                DONE?
1177   0869 26 EF             BNE SAV8
1178                  *
1179   086B 20 C9             BRA SAV7             KEEP GOING
1180                  *
1181   086D CE 08AC R SAV9    LDX #SAVFCB          POINT TO FCB
1182                  * WRITE OUT LAST RECORD HERE
1183   0870 86 02             LDA A #$02           HEADER BYTE
1184   0872 8D 2D             BSR PUTBYT
1185   0874 B6 0BDD R         LDA A SAVEX          HIGH-BYTE OF ADDRESS
1186   0877 8D 28             BSR PUTBYT
1187   0879 B6 0BDE R         LDA A SAVEX+1        LOW-BYTE OF ADDRESS
1188   087C 8D 23             BSR PUTBYT
1189   087E 17                TBA                  COUNT
1190   087F 8D 20             BSR PUTBYT
1191   0881 FE 0BDD R SAV10   LDX SAVEX            POINT TO MEMORY ADDRESS
1192   0884 A6 00             LDA A 0,X            GET BYTE
1193   0886 08                INX
1194   0887 FF 0BDD R         STX SAVEX
1195   088A CE 08AC R         LDX #SAVFCB          POINT TO FCB
1196   088D 8D 12             BSR PUTBYT
1197   088F 5A                DEC B                DONE?
1198   0890 26 EF             BNE SAV10
1199                  *
1200                          CLOSE                CLOSE SAVE-FILE
1201 +  0892 3F                SWI
1202 +  0893 15                FCB 21
1203   0894 6D 05             TST FCBSTA,X         CHECK STATUS
1204   0896 26 01             BNE SAVERR           BAD?
1205                  *
1206   0898 39                RTS                  RETURN TO CLI
1207                  *
1208   0899 CE 08AC R SAVERR  LDX #SAVFCB
1209                          PRTERR               PRINT ERROR MESSAGE
1210 +  089C 3F                SWI
1211 +  089D 1E                FCB 30
1212                          CLOSE                MUST CLOSE FILE
1213 +  089E 3F                SWI
1214 +  089F 15                FCB 21
1215   08A0 39                RTS
1216                  *
1217                  PUTBYT  WRITE                WRITE BYTE TO FILE
1218 +  08A1 3F                SWI
1219 +  08A2 19                FCB 25
1220   08A3 6D 05             TST FCBSTA,X         CHECK STATUS
1221   08A5 27 04             BEQ PUTB2            GOOD
1222                  *
1223   08A7 31                INS
1224   08A8 31                INS                  REMOVE SUB-RETURN
1225   08A9 20 EE             BRA SAVERR           ERROR AND QUIT
1226                  *
1227   08AB 39        PUTB2   RTS                  GOOD RETURN
1228                  *
1229   08AC 0002      SAVFCB  RMB 2
1230   08AE 44                FCC 'DSK'
1231   08B1 0002              RMB 2
1232   08B3 08D6    R         FDB SAVBUF
1233   08B5 0021              RMB 33
1234   08D6 0080      SAVBUF  RMB SECSIZ
1235                  *
1237                  * PROCESS 'LOAD' COMMAND
1238                  *
1239                  * LOAD MEMORY FROM BINARY OR COMMAND DISK FILE
1240                  * IF TRANSFER ADDRESS IN FILE (COMMAND TYPE), SAVE IT
1241                  * IN "VALUE" BASE-PAGE LOCATION
1242                  * OTHERWISE, "VALUE"=0
1243                  *
1244   0956 CE 08AC R LODCMD LDX #SAVFCB           POINT TO FCB
1245                          FMTFCB               FORMAT [DRIVE:] FILE.EXT
1246 +  0959 3F                SWI
1247 +  095A 2C                FCB 44
1248                          PRTERR               PRINT ERROR MESSAGES
1249 +  095B 3F                SWI
1250 +  095C 1E                FCB 30
1251   095D 6D 05             TST FCBSTA,X         ERROR?
1252   095F 26 04             BNE LOD5             IF SO, QUIT
1253                  *
1254                          LOADB                PERFORM LOAD BINARY
1255 +  0961 3F                SWI
1256 +  0962 25                FCB 37
1257                          PRTERR               PRINT ERROR MESSAGE
1258 +  0963 3F                SWI
1259 +  0964 1E                FCB 30
1260   0965 39        LOD5    RTS                  RETURN TO CLI
1261                  *
1262                  *
1263                  *
1264                  * LOAD-BINARY PROCESSING
1265                  *
1266   0966 7F 0027   @LOADB  CLR VALUE            NO XFER-ADDRESS YET
1267   0969 7F 0028           CLR VALUE+1
1268   096C 30                TSX
1269   096D EE 05             LDX UXH,X            POINT TO FCB
1270   096F 6F 06             CLR FCBDTT,X         INPUT
1271   0971 6F 29             CLR FCBSCF,X         NO COMPRESSION
1272   0973 6F 05             CLR FCBSTA,X         GOOD STATUS
1273                          OPEN                 OPEN SAVE-FILE
1274 +  0975 3F                SWI
1275 +  0976 14                FCB 20
1276   0977 6D 05             TST FCBSTA,X         CHECK STATUS
1277   0979 27 01             BEQ LOADB2           GOOD
1278                  *
1279   097B 39                RTS                  BAD, QUIT
1280                  *
1281   097C A6 1D     LOADB2  LDA A FCBTYP,X       CHECK FILE TYPE
1282   097E 27 22             BEQ LOADB3           (0) BINARY TYPE, OK
1283                  *
1284   0980 81 01             CMP A #1             (1) COMMAND TYPE, OK
1285   0982 27 1E             BEQ LOADB3
1286                  *
1287   0984 CE 098F R         LDX #TYPMSG          FILE-TYPE ERROR
1288                          PRTMSG
1289 +  0987 3F                SWI
1290 +  0988 31                FCB 49
1291   0989 30                TSX
1292   098A EE 05             LDX UXH,X            POINT TO FCB
1293                          CLOSE                CLOSE FILE
1294 +  098C 3F                SWI
```

```
1295 +  098D 15              FCB 21
1296    098E 39              RTS              RETURN
1297                 *
1298    098F 20      TYPMSG  FCC ' ILLEGAL FILE-TYPE'
1299    09A1 0D              FCB $0D
1300                 *
1301    09A2 30      LOADB3  TSX
1302    09A3 EE 05           LDX UXH,X        POINT TO FCB
1303    09A5 8D 39           BSR GETBYT       GET A BYTE FROM FILE
1304    09A7 81 16           CMP A #$16       XFER-ADDR. MARK?
1305    09A9 26 0A           BNE LOADB4       NO
1306                 *
1307                 * HANDLE TRANSFER ADDRESS HERE
1308                 *
1309    09AB 8D 33           BSR GETBYT       GET ADDRESS-HIGH
1310    09AD 97 27           STA A VALUE
1311    09AF 8D 2F           BSR GETBYT       GET ADDRESS-LOW
1312    09B1 97 28           STA A VALUE+1
1313    09B3 20 ED           BRA LOADB3       GET NEW FRAME
1314                 *
1315    09B5 81 02   LOADB4  CMP A #$02       FRAME HEADER MARK?
1316    09B7 27 03           BEQ LOADB5       YES
1317                 *
1318                 CLOSE           CLOSE FILE
1319 +  09B9 3F               SWI
1320 +  09BA 15               FCB 21
1321    09BB 39              RTS              DONE WITH LOAD
1322                 *
1323                 * HANDLE BINARY RECORD HERE
1324                 *
1325    09BC 8D 22   LOADB5  BSR GETBYT       GET ADDRESS-HIGH
1326    09BE B7 0BDD R       STA A SAVEX
1327    09C1 8D 1D           BSR GETBYT       GET ADDRESS-LOW
1328    09C3 B7 0BDE R       STA A SAVEX+1
1329    09C6 8D 18           BSR GETBYT       GET FRAME COUNT
1330    09C8 B7 0BE4 R       STA A SAVEA
1331                 *
1332    09CB 30      LOADB6  TSX
1333    09CC EE 05           LDX UXH,X        POINT TO FCB
1334    09CE 8D 10           BSR GETBYT       GET DATA BYTE
1335    09D0 FE 0BDD R       LDX SAVEX        GET ADDRESS
1336    09D3 A7 00           STA A 0,X        STORE DATA BYTE
1337    09D5 08              INX
1338    09D6 FF 0BDD R       STX SAVEX
1339    09D9 7A 0BE4 R       DEC SAVEA        COUNT DOWN FRAME
1340    09DC 26 ED           BNE LOADB6
1341                 *
1342    09DE 20 C2           BRA LOADB3       GET NEW FRAME
1343                 *
1344                 *
1345                 GETBYT  READ             READ BYTE FROM FILE
1346 +  09E0 3F               SWI
1347 +  09E1 18               FCB 24
1348    09E2 E6 05           LDA B FCBSTA,X   CHECK STATUS
1349    09E4 27 0A           BEQ GETB3        GOOD
1350                 *
1351    09E6 C1 08           CMP B #8         END-FILE?
1352    09E8 26 02           BNE GETB2        NO
1353                 *
1354    09EA 6F 05           CLR FCBSTA,X     END-FILE IS NOT ERROR HERE
1355                 *
1356    09EC 31      GETB2   INS              REMOVE SUB-RETURN
1357    09ED 31              INS              (WILL RETURN TO CALL)
1358                         CLOSE            CLOSE FILE
1359 +  09EE 3F               SWI
1360 +  09EF 15               FCB 21
1361                 *
1362    09F0 39      GETB3   RTS
1364                 * PROCESS 'RENAME' COMMAND
1365                 *
1366                 * SYNTAX: RENAME [DRIVE:] OLDNAME.EXT, NEWNAME.EXT
1367                 *
1368    09F1 CE 08AC R RENCMD LDX #SAVFCB             POINT TO FCB
1369    09F4 6F 06           CLR FCBDTT,X             INPUT
1370                         FMTFCB                   FORMAT [DRIVE:] FILE.EXT
1371 +  09F6 3F               SWI
1372 +  09F7 2C               FCB 44
1373                         PRTERR                   PRINT ERROR MESSAGES
1374 +  09F8 3F               SWI
1375 +  09F9 1E               FCB 30
1376    09FA 6D 05           TST FCBSTA,X             ERROR?
1377    09FC 26 28           BNE RENAM5               YES, QUIT
1378                 *
1379    09FE BD 0EFE R       JSR SFILE                SEARCH DIRECTORY
1380    0A01 CE 000C R       LDX #SYSFCB              GET DIRECTORY POINTERS
1381    0A04 A6 0A           LDA A FCBTRK,X           TRACK
1382    0A06 E6 0B           LDA B FCBSCT,X           SECTOR
1383    0A08 A7 1F           STA A FCBFTS,X
1384    0A0A E7 20           STA B FCBFTS+1,X
1385    0A0C A6 27           LDA A FCBIND,X           BLOCK POINTER
1386    0A0E E6 28           LDA B FCBIND+1,X
1387    0A10 A7 21           STA A FCBLTS,X
1388    0A12 E7 22           STA B FCBLTS+1,X
1389    0A14 EE 27           LDX FCBIND,X             GET DIRECTORY BLOCK POINTE
1390    0A16 A6 0E           LDA A FIBACS,X           CHECK ACCESS CODE
1391    0A18 81 02           CMP A #$02               RENAME-ABLE?
1392    0A1A 26 03           BNE *+5                  YES
1393                 *
1394    0A1C 7E 0AE3 R       JMP SECERR               NO, ERROR
1395                 *
1396    0A1F CE 08AC R       LDX #SAVFCB
1397    0A22 6D 05           TST FCBSTA,X             ERROR CHECK
1398    0A24 27 16           BEQ RENAM6               NO ERROR
1399                 *
1400    0A26 CE 0A2C R RENAM5 LDX #RENMSG             RENAME ERROR
1401                         PRTMSG
1402 +  0A29 3F               SWI
1403 +  0A2A 31               FCB 49
1404    0A2B 39              RTS
1405                 *
1406    0A2C 20      RENMSG  FCC ' RENAMING ERROR'
1407    0A3B 0D              FCB $0D
1408                 *
1409                 RENAM6  NXTOK                    GET TOKEN FROM CLI
1410 +  0A3C 3F               SWI
1411 +  0A3D 2F               FCB 47
1412    0A3E 96 26           LDA A CLASS
1413    0A40 81 04           CMP A #4                 DELIMITER?
1414    0A42 27 06           BEQ RENAM8               YES
1415                 *
1416    0A44 CE 03AA R RENAM7 LDX #FORMAT             NO, FORMAT ERROR
1417                         PRTMSG
```

```
0A47 3F              SWI
0A48 31              FCB 49
0A49 39              RTS
                *
                RENAM8 NXTOK                 GET TOKEN FROM CLI
0A4A 3F              SWI
0A4B 2F              FCB 47
0A4C D6 25           LDA B RC                CHECK RC
0A4E C1 01           CMP B #1                UNAMBIG. NAME?
0A50 26 F2           BNE RENAM7              NO, ERROR
                *
0A52 DE 20           LDX DESCRA              GET NAME POINTER
0A54 FF 0BDD R       STX SAVEX               SAVE IT
0A57 96 22           LDA A DESCRC            GET LENGTH
0A59 B7 0BE4 R       STA A SAVEA             SAVE IT
                     NXTOK                   GET TOKEN FROM CLI
0A5C 3F              SWI
0A5D 2F              FCB 47
0A5E D6 25           LDA B RC                CHECK RC
0A60 C1 2E           CMP B #'.               PERIOD?
0A62 26 E0           BNE RENAM7              NO, ERROR
                *
0A64 7C 0BE4 R       INC SAVEA               COUNT PERIOD
                     NXTOK                   GET TOKEN FROM CLI
0A67 3F              SWI
0A68 2F              FCB 47
0A69 D6 25           LDA B RC                CHECK RC
0A6B C1 01           CMP B #1                UNAMBIG. NAME?
0A6D 26 D5           BNE RENAM7              NO, ERROR
                *
0A6F D6 22           LDA B DESCRC            GET LENGTH OF EXT
0A71 FB 0BE4 R       ADD B SAVEA             TOTAL LENGTH
0A74 CE 08BC R       LDX #SAVFCB+FCBNAM      POINT TO NAME FIELD IN FCB
                     PSHX
0A77 3F              SWI
0A78 05              FCB 5
0A79 FE 0BDD R       LDX SAVEX               POINT TO NAME IN CLI
                     PSHX
0A7C 3F              SWI
0A7D 05              FCB 5
                     FMTS                    FORMAT NAME INTO FCB
0A7E 3F              SWI
0A7F 34              FCB 52
0A80 31              INS
0A81 31              INS                     CLEAN STACK
0A82 31              INS
0A83 31              INS
0A84 5D              TST B                   ERROR?
0A85 26 BD           BNE RENAM7              YES
                *
0A87 CE 08AC R       LDX #SAVFCB
0A8A BD 0EFE R       JSR SFILE               SEARCH DIRECTORY
0A8D CE 08AC R       LDX #SAVFCB
0A90 A6 05           LDA A FCBSTA,X          CHECK STATUS
0A92 26 03           BNE *+5                 NOT FOUND?
                *
0A94 7E 0ACD R       JMP RENAM9              YES, DUPLICATE NAME
                *
0A97 81 01           CMP A #1                NOT FOUND (GOOD)?
0A99 27 03           BEQ *+5                 YES
                *
```

```
0A9B 7E 0A26 R RENM5  JMP RENAM5             NO, ERROR
                *
0A9E CE 000C R        LDX #SYSFCB            POINT TO DIRECTORY FCB
0AA1 6F 05            CLR FCBSTA,X
0AA3 A6 1F            LDA A FCBFTS,X         SET UP TO SAVED T/S
0AA5 E6 20            LDA B FCBFTS+1,X
0AA7 A7 0A            STA A FCBTRK,X
0AA9 E7 0B            STA B FCBSCT,X
                      IOHDR                  READ DIRECTORY SECTOR
0AAB 3F               SWI
0AAC 13               FCB 19
0AAD 4D               TST A                  ERROR?
0AAE 26 EB            BNE RENM5              YES
                *
0AB0 EE 21            LDX FCBLTS,X           POINT TO OLD NAME IN SECTOR
                      PSHX
0AB2 3F               SWI
0AB3 05               FCB 5
0AB4 CE 08BC R        LDX #SAVFCB+FCBNAM     POINT TO NEW NAME
                      PSHX
0AB7 3F               SWI
0AB8 05               FCB 5
0AB9 C6 0C            LDA B #12
                      MOVC                   MOVE NAME INTO DIRECTORY
0ABB 3F               SWI
0ABC 11               FCB 17
0ABD 31               INS
0ABE 31               INS
0ABF 31               INS                    CLEAN STACK
0AC0 31               INS
0AC1 CE 000C R        LDX #SYSFCB            POINT TO SYSTEM FCB
0AC4 86 FF            LDA A #$FF
0AC6 A7 06            STA A FCBDTT,X         MAKE 'OUTPUT'
                      IOHDR                  WRITE NEW NAME INTO DIRECTORY
0AC8 3F               SWI
0AC9 13               FCB 19
0ACA 6F 06            CLR FCBDTT,X           MAKE 'INPUT' AGAIN
0ACC 39               RTS
                *
0ACD CE 0AD3 R RENAM9 LDX #DUPERR            DUPLICATE NAME ERROR
                      PRTMSG
0AD0 3F               SWI
0AD1 31               FCB 49
0AD2 39               RTS
                *
0AD3 20         DUPERR FCC ' DUPLICATE NAME'
0AE2 0D               FCB $0D
                *
0AE3 CE 0AE9 R SECERR LDX #SECURE            SECURITY ERROR
                      PRTMSG
0AE6 3F               SWI
0AE7 31               FCB 49
0AE8 39               RTS
                *
0AE9 20         SECURE FCC ' SECURITY ERROR'
0AF8 0D               FCB $0D
                * PROCESS 'INITIALIZE' COMMAND
                * CALL TRANSIENT FILE TO PROCESS THIS COMMAND
                *
                INICMD NXTOK          GET NEXT TOKEN FROM COMMAND LINE
0AF9 3F               SWI
```

```
1541 +  0AFA 2F                FCB 47
1542    0AFB D6 25             LDA B RC        CHECK RC
1543    0AFD C1 03             CMP B #3        NUMBER?
1544    0AFF 27 0C             BEQ INICD2      YES
1545                   *
1546    0B01 96 26             LDA A CLASS     CHECK CLASS OF TOKEN
1547    0B03 81 04             CMP A #4        DELIMITER?
1548    0B05 27 F2             BEQ INICMD      IF SO, PARSE AGAIN
1549                   *
1550    0B07 CE 03AA R         LDX #FORMAT     IF NOT, PRINT ERROR MESSAGE
1551                           PRTMSG
1552 +  0B0A 3F                 SWI
1553 +  0B0B 31                 FCB 49
1554    0B0C 39                RTS             RETURN TO CLI
1555                   *
1556    0B0D 7D 0027   INICD2  TST VALUE       CHECK NUMBER
1557    0B10 26 20             BNE INIERR      BAD (>255)
1558                   *
1559    0B12 96 28             LDA A VALUE+1   CHECK NUMBER
1560    0B14 81 03             CMP A #3        (4 DRIVES)
1561    0B16 22 1A             BHI INIERR      BAD
1562                   *
1563    0B18 36                PSH A           SAVE NUMBER
1564    0B19 CE 0B38 R         LDX #INITLN
1565    0B1C DF 23             STX CUCHAR      SET UP DUMMY CLI
1566    0B1E BD 0956 R         JSR LODCMD      ISSUE 'LOAD' COMMAND FOR TRANSIENT
1567    0B21 32                PUL A           RESTORE NUMBER
1568    0B22 CE 08AC R         LDX #SAVFCB
1569    0B25 6D 05             TST FCBSTA,X    CHECK FOR ERRORS
1570    0B27 26 08             BNE INICD3      YES, QUIT
1571                   *
1572    0B29 DE 27             LDX VALUE       IS THERE XFER-ADDRESS?
1573    0B2B 27 04             BEQ INICD3      NO, QUIT
1574                   *
1575    0B2D 97 28             STA A VALUE+1   PASS DRIVE NO.  TO TRANSIENT
1576    0B2F 6E 00             JMP 0,X         START IT UP
1577                   *
1578    0B31 39        INICD3 RTS             DONE!
1579                   *
1580    0B32 CE 03B7 R INIERR LDX #NUMBER     PRINT ERROR MESSAGE
1581                           PRTMSG
1582 +  0B35 3F                 SWI
1583 +  0B36 31                 FCB 49
1584    0B37 39                RTS             RETURN TO CLI
1585                   *
1586    0B38 30        INITLN FCC '0:INIT.CMD'
1587    0B42 0D                FCB $0D
1589                   * PROCESS 'DIRECTORY' COMMAND
1590                   *
1591    0B43 44        DIRHDR FCC 'DIRECTORY OF DRIVE '
1592    0B56 0001      DIRDRV RMB 1
1593    0B57 0A0D              FDB $0A0D
1594                   *
1595    0B59 20        DIRFLD FCC '    NAME       T   A  FT-FS  LT-LS   NS'
1596    0B80 0A0D              FDB $0A0D
1597                   *
1598    0B82 0028      DIRLIN RMB 40          DIRECTORY LINE BUFFER
1599    0BAA 0D                FCB $0D
1600                   *
1601    0BAB 0002      NSEC   RMB 2           NUMBER OF SECTORS USED
1602                   *
1603    0BAD 000D      BUFLIN RMB 13          STORAGE FOR FORMATTED FILE NAME
1604                   *
1605    0BBA 2A        ALLFIL FCC '*.*'       DEFAULT FILE NAME
1606                   *
1607    0BBD 0A        DIRLN2 FCB $0A
1608    0BBE 0005              RMB 5           SPACE FOR COUNT OF SECTORS
1609    0BC3 2E                FCC '. SECTORS USED'
1610    0BD1 0A0D              FDB $0A0D
1611                   *
1612    0BD3 2710      KIOK   FDB 10000       POWERS-OF-10 TABLE
1613    0BD5 03E8              FDB 1000
1614    0BD7 0064              FDB 100
1615    0BD9 000A              FDB 10
1616    0BDB 0001              FDB 1
1617                   *
1618    0BDD 0002      SAVEX  RMB 2           TEMP.  STORAGE
1619    0BDF 0002      SAVEX1 RMB 2
1620    0BE1 0002      TMPX   RMB 2
1621    0BE3 0001      LPTFLG RMB 1
1622    0BE4 0001      SAVEA  RMB 1
1623                   *
1624                   *
1625                   *
1626    0BE5 7F 0BE4 R DIRCMD CLR SAVEA       DEFAULT DRIVE=0
1627    0BE8 7F 0BE3 R         CLR LPTFLG      DEFAULT TO CONSOLE
1628    0BEB 86 0C             LDA A #12       DEFAULT NAME HAS 12 CHARS.
1629    0BED B7 0BDD R         STA A SAVEX     SAVE IT
1630    0BF0 CE 0BAD R         LDX #BUFLIN
1631                           PSHX
1632 +  0BF3 3F                 SWI
1633 +  0BF4 05                 FCB 5
1634    0BF5 CE 0BBA R         LDX #ALLFIL     DEFAULT FILE NAME=*.*
1635                           PSHX
1636 +  0BF8 3F                 SWI
1637 +  0BF9 05                 FCB 5
1638    0BFA C6 03             LDA B #3
1639                           FMTS            FORMAT *.* INTO BUFLIN
1640 +  0BFC 3F                 SWI
1641 +  0BFD 34                 FCB 52
1642    0BFE 31                INS
1643    0BFF 31                INS             CLEAN STACK
1644    0C00 31                INS
1645    0C01 31                INS
1646                   DIRCDO NXTOK           GET TOKEN FROM CLI
1647 +  0C02 3F                 SWI
1648 +  0C03 2F                 FCB 47
1649    0C04 D6 25             LDA B RC        CHECK RC
1650    0C06 C1 0D             CMP B #$0D      END OF LINE?
1651    0C08 26 03             BNE *+5         NO
1652                   *
1653    0C0A 7E 0C94 R         JMP DIRCD3      YES, USE DEFAULTS
1654                   *
1655    0C0D C1 2F             CMP B #'/       SWITCH INDICATOR?
1656    0C0F 26 0F             BNE DIRCD1      NO
1657                   *
1658                           NXTOK           GET SWITCH
1659 +  0C11 3F                 SWI
1660 +  0C12 2F                 FCB 47
1661    0C13 DE 20             LDX DESCRA
1662    0C15 A6 00             LDA A 0,X
1663    0C17 81 4C             CMP A #'L       LINE-PRINTER ('L')?
```

```
0C19 26 37            BNE DIR3      NO, FORMAT ERROR
                      *
0C1B 7C 0BE3 R        INC LPTFLG    YES, SET FLAG
0C1E 20 E2            BRA DIRCD0    GET NEW TOKEN
                      *
0C20 C1 03     DIRCD1 CMP B #3      TOKEN=NUMBER?
0C22 26 26            BNE DIR2      NO
                      *
0C24 7D 0027          TST VALUE     CHECK FOR VALID DRIVE NO.
0C27 26 0B            BNE DIR1      BAD?
                      *
0C29 96 28            LDA A VALUE+1 CHECK DRIVE NO.
0C2B 81 03            CMP A #3      DRIVE NO. 0,1,2,3
0C2D 22 05            BHI DIR1      BAD?
                      *
0C2F B7 0BE4 R        STA A SAVEA   SAVE DRIVE NO.
0C32 20 06            BRA DIR1A
                      *
0C34 CE 03B7 R DIR1   LDX #NUMBER   NUMBER ERROR
                      PRTMSG
 0C37 3F               SWI
 0C38 31               FCB 49
0C39 39               RTS           RETURN TO CLI
                      *
               DIR1A  NXTOK         GET NEXT TOKEN FROM CLI
 0C3A 3F               SWI
 0C3B 2F               FCB 47
0C3C D6 25            LDA B RC      CHECK RC
0C3E C1 0D            CMP B #$0D    END OF LINE?
0C40 27 52            BEQ DIRCD3    YES, USE DEFAULT FILE NAME
                      *
0C42 C1 3A            CMP B #':     COLON?
0C44 26 EE            BNE DIR1      IF NOT, BAD DRIVE NO.
                      *
                      NXTOK         GET NEXT TOKEN FROM CLI
 0C46 3F               SWI
 0C47 2F               FCB 47
0C48 D6 25            LDA B RC      CHECK RC
                      *
0C4A C1 01     DIR2   CMP B #1      NAME?
0C4C 27 0A            BEQ DIR4      YES
                      *
0C4E C1 02            CMP B #2      WILD-CARD NAME?
0C50 27 06            BEQ DIR4      YES
                      *
0C52 CE 03AA R DIR3   LDX #FORMAT   OTHERWISE FORMAT ERROR
                      PRTMSG
 0C55 3F               SWI
 0C56 31               FCB 49
0C57 39               RTS           RETURN TO CLI
                      *
0C58 DE 20     DIR4   LDX DESCRA    POINT TO NAME
0C5A FF 0BDF R        STX SAVEX1    SAVE POINTER
0C5D 96 22            LDA A DESCRC  GET LENGTH
0C5F B7 0BDD R        STA A SAVEX   SAVE IT
                      NXTOK         GET NEXT TOKEN FROM CLI
 0C62 3F               SWI
 0C63 2F               FCB 47
0C64 D6 25            LDA B RC      CHECK RC
0C66 C1 2E            CMP B #'.     PERIOD?
0C68 26 E8            BNE DIR3      NO, ERROR
                      *
0C6A 7C 0BDD R DIR5   INC SAVEX     COUNT PERIOD
                      NXTOK         GET NEXT TOKEN FROM CLI
 0C6D 3F               SWI
 0C6E 2F               FCB 47
0C6F D6 25            LDA B RC      CHECK RC
0C71 C1 01            CMP B #1      NAME?
0C73 27 06            BEQ DIR6      YES
                      *
0C75 C1 02            CMP B #2      WILD-CARD NAME?
0C77 27 02            BEQ DIR6      YES
                      *
0C79 20 D7            BRA DIR3      FORMAT ERROR
                      *
0C7B D6 22     DIR6   LDA B DESCRC
0C7D FB 0BDD R        ADD B SAVEX   GET TOTAL LENGTH
0C80 CE 0BAD R        LDX #BUFLIN
                      PSHX
 0C83 3F               SWI
 0C84 05               FCB 5
0C85 FE 0BDF R        LDX SAVEX1
                      PSHX
 0C88 3F               SWI
 0C89 05               FCB 5
                      FMTS          FORMAT NAME INTO BUFLIN
 0C8A 3F               SWI
 0C8B 34               FCB 52
0C8C 31               INS
0C8D 31               INS           CLEAN STACK
0C8E 31               INS
0C8F 31               INS
0C90 C1 02            CMP B #2      BAD NAME?
0C92 27 BE            BEQ DIR3      FORMAT ERROR
                      *
0C94 7F 0BAB R DIRCD3 CLR NSEC
0C97 7F 0BAC R        CLR NSEC+1    NO USED SECTORS YET
0C9A 7D 0BE3 R        TST LPTFLG    WANT LINE-PRINTER?
0C9D 27 0F            BEQ DIR7      NO
                      *
0C9F CE 0298 R        LDX #CONRCB   MAKE CONSOLE INTO 'LPT'
0CA2 86 4C            LDA A #'L
0CA4 A7 02            STA A RCBGDT,X
0CA6 86 50            LDA A #'P
0CA8 A7 03            STA A RCBGDT+1,X
0CAA 86 54            LDA A #'T
0CAC A7 04            STA A RCBGDT+2,X
                      *
0CAE B6 0BE4 R DIR7   LDA A SAVEA   GET DRIVE NO.
0CB1 8B 30            ADD A #$30    MAKE ASCII
0CB3 B7 0B56 R        STA A DIRDRV
0CB6 CE 0B43 R        LDX #DIRHDR   PRINT HEADER LINE 1
                      PRTMSG
 0CB9 3F               SWI
 0CBA 31               FCB 49
0CBB CE 0B59 R        LDX #DIRFLD   PRINT HEADER LINE 2
                      PRTMSG
 0CBE 3F               SWI
 0CBF 31               FCB 49
0CC0 CE 000C R        LDX #SYSFCB   POINT TO SYSTEM FCB
                      TXAB
 0CC3 3F               SWI
```

```
0CC4 02                 FCB 2
0CC5 CE 0036 R          LDX #BUFFER   PUT BUFFER ADDRESS IN
                        XABX
0CC8 3F                 SWI
0CC9 04                 FCB 4
0CCA A7 07              STA A FCBDBA,X
0CCC E7 08              STA B FCBDBA+1,X
0CCE B6 0BE4 R          LDA A SAVEA
0CD1 A7 09              STA A FCBDRV,X      PUT DRIVE NO. IN
                        OPEND               OPEN DIRECTORY
0CD3 3F                 SWI
0CD4 17                 FCB 23
                *
0CD5 A6 05      DIRCD4  LDA A FCBSTA,X      CHECK STATUS
0CD7 27 4B              BEQ DIRLST          VALID ENTRY?
                *
0CD9 81 01              CMP A #1            END-OF-DIRECTORY?
0CDB 26 44              BNE DIRERR          IF NOT, ERROR
                *
0CDD B6 0BAB R          LDA A NSEC          OUTPUT NUMBER OF SECTORS USED
0CE0 F6 0BAC R          LDA B NSEC+1
0CE3 CE 0BBE R          LDX #DIRLN2+1
                *
                * CONVERT BINARY (16 BITS) TO 5 DECIMAL CHARS.
                * (A,B)=BINARY VALUE
                * (X)= ADDRESS TO PLACE CHARS IN ASCII
                *
0CE6 FF 0BDD R  CVBTD   STX SAVEX       SAVE DATA POINTER
0CE9 CE 0BD3 R          LDX #KIOK       POINT TO CONSTANT TABLE
0CEC 7F 0BE4 R  CVDEC1  CLR SAVEA       INIT. CHAR
0CEF E0 01      CVDEC2  SUB B 1,X
0CF1 A2 00              SBC A 0,X
0CF3 25 05              BCS CVDEC5      OVERFLOW?
                *
0CF5 7C 0BE4 R          INC SAVEA       NO, BUMP CHAR.
0CF8 20 F5              BRA CVDEC2
                *
0CFA EB 01      CVDEC5  ADD B 1,X       RESTORE PARTIAL RESULT
0CFC A9 00              ADC A 0,X
0CFE 36                 PSH A
0CFF FF 0BDF R          STX SAVEX1      SAVE REGISTERS
0D02 FE 0BDD R          LDX SAVEX       GET POINTER
0D05 B6 0BE4 R          LDA A SAVEA     GET CHAR.
0D08 8B 30              ADD A #$30      MAKE ASCII
0D0A A7 00              STA A 0,X       STORE CHAR.
0D0C 32                 PUL A
0D0D 08                 INX
0D0E FF 0BDD R          STX SAVEX       SAVE POINTER
0D11 FE 0BDF R          LDX SAVEX1      RECOVER REGISTER
0D14 08                 INX             MOVE TO NEXT TABLE ENTRY
0D15 08                 INX
0D16 8C 0BDD R          CPX #KIOK+10    DONE?
0D19 26 D1              BNE CVDEC1      NO
                *
0D1B CE 0BBD R          LDX #DIRLN2             PRINT END LINE
                        PRTMSG
0D1E 3F                 SWI
0D1F 31                 FCB 49
0D20 39                 RTS                     DONE!!.
                *
                DIRERR  PRTERR                  PRINT ERROR MESSAGE
0D21 3F                 SWI
0D22 1E                 FCB 30
0D23 39                 RTS                     RETURN TO CLI
                *
                * OUTPUT FORMATTED DIRECTORY RECORD HERE
                *
0D24 EE 27      DIRLST  LDX FCBIND,X            GET FIRST CHAR. OF BLOCK
0D26 A6 00              LDA A 0,X
0D28 81 20              CMP A #$20              FIRST CHAR=BLANK?
0D2A 26 03              BNE *+5                 NO
                *
0D2C 7E 0E31 R          JMP DIRNXT              GET NEXT ENTRY
                *
                        PSHX
0D2F 3F                 SWI
0D30 05                 FCB 5
0D31 CE 0BAD R          LDX #BUFLIN
                        PSHX
0D34 3F                 SWI
0D35 05                 FCB 5
0D36 C6 0C              LDA B #12
                        CMWC                    COMPARE NAMES (WITH WILD-CARDS)
0D38 3F                 SWI
0D39 35                 FCB 53
0D3A 31                 INS
0D3B 31                 INS
0D3C 31                 INS
0D3D 31                 INS
0D3E 27 03              BEQ *+5                 NAMES MATCH
                *
0D40 7E 0E31 R          JMP DIRNXT              NO MATCH, GET NEXT NAME
                *
0D43 CE 0BA9 R          LDX #DIRLIN+39
0D46 C6 20              LDA B #$20              BLANK OUT DIRECTORY LINE
0D48 86 27              LDA A #39
0D4A E7 00      DIRCD5  STA B 0,X
0D4C 09                 DEX
0D4D 4A                 DEC A
0D4E 26 FA              BNE DIRCD5
                *
                        PSHX                    STACK 'TO' ADDRESS
0D50 3F                 SWI
0D51 05                 FCB 5
0D52 CE 000C R          LDX #SYSFCB
0D55 EE 27              LDX FCBIND,X
                        PSHX                    STACK 'FROM' ADDRESS
0D57 3F                 SWI
0D58 05                 FCB 5
0D59 C6 0C              LDA B #12               SET UP FOR MOVE (FROM DATA BLOCK
                        MOVC                    TO DIRECTORY LINE, 12 CHARACTERS
0D5B 3F                 SWI
0D5C 11                 FCB 17
                        PULX
0D5D 3F                 SWI
0D5E 06                 FCB 6
                        PULX                    RECOVER LINE POINTER
0D5F 3F                 SWI
0D60 06                 FCB 6
0D61 08                 INX
0D62 08                 INX
0D63 08                 INX                     POINT TO LINE (15)
```

```
                                PSHX
0D64 3F                          SWI
0D65 05                          FCB 5
0D66 CE 000C R                  LDX #SYSFCB         POINT TO FCB
0D69 EE 27                      LDX FCBIND,X        POINT TO DATA BLOCK
0D6B A6 0D                      LDA A FIBTYP,X      GET FILE TYPE
0D6D 16                         TAB
0D6E BD 0E39 R                  JSR OUTHL           CONVERT TO ASCII
                                PULX
0D71 3F                          SWI
0D72 06                          FCB 6
0D73 A7 00                      STA A 0,X           PUT INTO LINE
0D75 08                         INX
0D76 17                         TBA
0D77 BD 0E3D R                  JSR OUTHR
0D7A A7 00                      STA A 0,X
0D7C 08                         INX
0D7D 08                         INX
0D7E 08                         INX                 POINT TO LINE (19)
                                PSHX
0D7F 3F                          SWI
0D80 05                          FCB 5
0D81 CE 000C R                  LDX #SYSFCB
0D84 EE 27                      LDX FCBIND,X
0D86 A6 0E                      LDA A FIBACS,X      GET FILE ACCESS CODE
0D88 16                         TAB
0D89 BD 0E39 R                  JSR OUTHL           CONVERT TO ASCII
                                PULX
0D8C 3F                          SWI
0D8D 06                          FCB 6
0D8E A7 00                      STA A 0,X           PUT INTO LINE
0D90 08                         INX
0D91 17                         TBA
0D92 BD 0E3D R                  JSR OUTHR
0D95 A7 00                      STA A 0,X
0D97 08                         INX
0D98 08                         INX
0D99 08                         INX                 POINT TO LINE (23)
                                PSHX
0D9A 3F                          SWI
0D9B 05                          FCB 5
0D9C CE 000C R                  LDX #SYSFCB
0D9F EE 27                      LDX FCBIND,X
0DA1 A6 0F                      LDA A FIBFTS,X      GET FIRST TRACK
0DA3 16                         TAB
0DA4 BD 0E39 R                  JSR OUTHL           CONVERT HIGH NIBBLE
                                PULX
0DA7 3F                          SWI
0DA8 06                          FCB 6
0DA9 A7 00                      STA A 0,X           PUT INTO LINE
0DAB 08                         INX                 POINT TO LINE (24)
0DAC 17                         TBA
0DAD BD 0E3D R                  JSR OUTHR           CONVERT LOW NIBBLE
0DB0 A7 00                      STA A 0,X           PUT INTO LINE
0DB2 08                         INX
0DB3 08                         INX                 POINT TO LINE (26)
                                PSHX
0DB4 3F                          SWI
0DB5 05                          FCB 5
0DB6 CE 000C R                  LDX #SYSFCB
0DB9 EE 27                      LDX FCBIND,X
```

```
0DBB A6 10                      LDA A FIBFTS+1,X    GET FIRST SECTOR
0DBD 16                         TAB
0DBE BD 0E39 R                  JSR OUTHL           CONVERT HIGH NIBBLE
                                PULX
0DC1 3F                          SWI
0DC2 06                          FCB 6
0DC3 A7 00                      STA A 0,X           PUT INTO LINE
0DC5 08                         INX                 POINT TO LINE (27)
0DC6 17                         TBA
0DC7 BD 0E3D R                  JSR OUTHR           CONVERT LOW NIBBLE
0DCA A7 00                      STA A 0,X           PUT INTO LINE
0DCC 08                         INX
0DCD 08                         INX
0DCE 08                         INX                 POINT TO LINE (30)
                                PSHX
0DCF 3F                          SWI
0DD0 05                          FCB 5
0DD1 CE 000C R                  LDX #SYSFCB
0DD4 EE 27                      LDX FCBIND,X
0DD6 A6 11                      LDA A FIBLTS,X      GET LAST TRACK
0DD8 16                         TAB
0DD9 8D 5E                      BSR OUTHL           CONVERT HIGH NIBBLE
                                PULX
0DDB 3F                          SWI
0DDC 06                          FCB 6
0DDD A7 00                      STA A 0,X           PUT INTO LINE
0DDF 08                         INX                 POINT TO LINE (31)
0DE0 17                         TBA
0DE1 8D 5A                      BSR OUTHR           CONVERT LOW NIBBLE
0DE3 A7 00                      STA A 0,X           PUT INTO LINE
0DE5 08                         INX
0DE6 08                         INX                 POINT TO LINE (33)
                                PSHX
0DE7 3F                          SWI
0DE8 05                          FCB 5
0DE9 CE 000C R                  LDX #SYSFCB
0DEC EE 27                      LDX FCBIND,X
0DEE A6 12                      LDA A FIBLTS+1,X    GET LAST SECTOR
0DF0 16                         TAB
0DF1 8D 46                      BSR OUTHL           CONVERT HIGH NIBBLE
                                PULX
0DF3 3F                          SWI
0DF4 06                          FCB 6
0DF5 A7 00                      STA A 0,X           PUT INTO LINE
0DF7 08                         INX                 POINT TO LINE (34)
0DF8 17                         TBA
0DF9 8D 42                      BSR OUTHR           CONVERT LOW NIBBLE
0DFB A7 00                      STA A 0,X           PUT INTO LINE
0DFD 08                         INX
0DFE 08                         INX
0DFF 08                         INX                 POINT TO LINE (37)
                                PSHX
0E00 3F                          SWI
0E01 05                          FCB 5
0E02 CE 000C R                  LDX #SYSFCB
0E05 EE 27                      LDX FCBIND,X
0E07 EE 13                      LDX FIBNMS,X        GET SECTOR COUNT
                                TXAB
0E09 3F                          SWI
0E0A 02                          FCB 2
0E0B FE 0BAB R                  LDX NSEC
```

```
2030                            ADDABX              ADD NSEC TO COUNT
2031 +  0E0E 3F                 SWI
2032 +  0E0F 08                 FCB 8
2033    0E10 FF 0BAB R          STX NSEC
2034                            PULX
2035 +  0E13 3F                 SWI
2036 +  0E14 06                 FCB 6
2037    0E15 36                 PSH A               SAVE 'A'
2038    0E16 8D 21              BSR OUTHL           CONVERT HIGH NIBBLE
2039    0E18 A7 00              STA A 0,X           PUT INTO LINE
2040    0E1A 08                 INX                 POINT TO LINE (38)
2041    0E1B 32                 PUL A               RESTORE 'A'
2042    0E1C 8D 1F              BSR OUTHR           CONVERT LOW NIBBLE
2043    0E1E A7 00              STA A 0,X           PUT INTO LINE
2044    0E20 08                 INX                 POINT TO LINE (39)
2045    0E21 17                 TBA
2046    0E22 8D 15              BSR OUTHL           CONVERT HIGH NIBBLE (LOW BYTE)
2047    0E24 A7 00              STA A 0,X           PUT INTO LINE
2048    0E26 08                 INX                 POINT TO LINE (40)
2049    0E27 17                 TBA
2050    0E28 8D 13              BSR OUTHR           CONVERT LOW NIBBLE (LOW BYTE)
2051    0E2A A7 00              STA A 0,X           PUT INTO LINE
2052    0E2C CE 0B82 R          LDX #DIRLIN         PRINT LINE OF DIRECTORY
2053                            PRTMSG
2054 +  0E2F 3F                 SWI
2055 +  0E30 31                 FCB 49
2056                    *
2057    0E31 CE 000C R DIRNXT   LDX #SYSFCB         POINT TO FCB
2058                            GETDR               GET NEW DIRECTORY BLOCK
2059 +  0E34 3F                 SWI
2060 +  0E35 1A                 FCB 26
2061    0E36 7E 0CD5 R          JMP DIRCD4          CONTINUE
2062                    *
2063    0E39 44         OUTHL   LSR A               CONVERT LEFT NIBBLE TO ASCII
2064    0E3A 44                 LSR A
2065    0E3B 44                 LSR A
2066    0E3C 44                 LSR A
2067                    *
2068    0E3D 84 0F      OUTHR   AND A #$0F          CONVERT RIGHT NIBBLE TO ASCII
2069    0E3F 8B 30              ADD A #$30
2070    0E41 81 39              CMP A #$39
2071    0E43 23 02              BLS *+4
2072                    *
2073    0E45 8B 07              ADD A #$07
2074                    *
2075    0E47 39                 RTS
2076                    *
2078                    * LOAD AND RUN A TRANSIENT FILE
2079                    * INDEX (STACKED) POINTS TO FCB WITH FILENAME AND DRIVE
2080                    *
2081    0E48 CE 08AC R @CHAIN   LDX #SAVFCB         MOVE DATA TO SYSTEM FCB
2082                            PSHX
2083 +  0E4B 3F                 SWI
2084 +  0E4C 05                 FCB 5
2085    0E4D 30                 TSX
2086    0E4E EE 07              LDX UXH+2,X         PASSED FCB ADDRESS
2087                            PSHX
2088 +  0E50 3F                 SWI
2089 +  0E51 05                 FCB 5
2090    0E52 C6 1E              LDA B #30           MOVE 30 CHARACTERS
2091                            MOVC
2092 +  0E54 3F                 SWI
2093 +  0E55 11                 FCB 17
2094    0E56 31                 INS
2095    0E57 31                 INS                 CLEAN STACK
2096    0E58 31                 INS
2097    0E59 31                 INS
2098    0E5A CE 08AC R          LDX #SAVFCB
2099                            TXAB
2100 +  0E5D 3F                 SWI
2101 +  0E5E 02                 FCB 2
2102    0E5F CE 08D6 R          LDX #SAVBUF         USE SYSTEM BUFFER
2103                            XABX
2104 +  0E62 3F                 SWI
2105 +  0E63 04                 FCB 4
2106    0E64 A7 07              STA A FCBDBA,X
2107    0E66 E7 08              STA B FCBDBA+1,X
2108    0E68 6F 05              CLR FCBSTA,X
2109                            LOADB               LOAD BINARY FILE
2110 +  0E6A 3F                 SWI
2111 +  0E6B 25                 FCB 37
2112    0E6C 6D 05              TST FCBSTA,X        CHECK STATUS
2113    0E6E 26 0F              BNE CHANER          ERROR?
2114                    *
2115    0E70 DE 27              LDX VALUE           TRANSFER ADDRESS?
2116    0E72 27 0B              BEQ CHANER          IF NOT, ERROR
2117                    *
2118    0E74 31                 INS
2119    0E75 31                 INS
2120    0E76 31                 INS                 CLEAN STACK (9 BYTES)
2121    0E77 31                 INS
2122    0E78 31                 INS                 7 BYTES FROM SWI
2123    0E79 31                 INS                 2 BYTES FROM JSR
2124    0E7A 31                 INS
2125    0E7B 31                 INS
2126    0E7C 31                 INS
2127    0E7D 6E 00              JMP 0,X             GO TO TRANS. ADDRESS
2128                    *
2129    0E7F CE 0EB7 R CHANER   LDX #CHANME         CHAIN ERROR
2130                            PSHX
2131 +  0E82 3F                 SWI
2132 +  0E83 05                 FCB 5
2133    0E84 CE 08BC R          LDX #SAVFCB+FCBNAM  NAME OF FILE
2134                            PSHX
2135 +  0E87 3F                 SWI
2136 +  0E88 05                 FCB 5
2137    0E89 C6 0C              LDA B #12           12 CHARACTERS
2138                            MOVC
2139 +  0E8B 3F                 SWI
2140 +  0E8C 11                 FCB 17
2141    0E8D 31                 INS
2142    0E8E 31                 INS                 CLEAN STACK
2143    0E8F 31                 INS
2144    0E90 31                 INS
2145    0E91 CE 0EA5 R          LDX #CHANLN
2146                            PRTMSG
2147 +  0E94 3F                 SWI
2148 +  0E95 31                 FCB 49
2149    0E96 31                 INS
2150    0E97 31                 INS
2151    0E98 31                 INS
2152    0E99 31                 INS                 CLEAN STACK (SWI+JSR)
```

```
0E9A 31              INS
0E9B 31              INS
0E9C 31              INS
0E9D 31              INS
0E9E 31              INS
0E9F CE 08AC R       LDX #SAVFCB
                     CLOSE                CLOSE FILE
 0EA2 3F              SWI
 0EA3 15              FCB 21
0EA4 39              RTS                  QUIT--WILL GO TO CLI
                *
0EA5 20         CHANLN FCC ' UNABLE TO CHAIN: '
0EB7 000C       CHANME RMB 12
0EC3 0A0D            FDB $0A0D
                * SEARCH DIRECTORY FILE FOR EMPTY SLOT
                * USES SYSTEM FCB AND BUFFER
                * PASS DRIVE NO. IN 'A' REGISTER
                * RETURNS TRACK, SECTOR OF SLOT IN FCBTRK,FCBSCT
                * RETURNS ADDRESS OF DIR. SLOT IN FCBIND
                * RETURNS ERROR STATUS IN FCBSTA
                *       0=FOUND SLOT
                *       1=NO AVAILABLE SLOT
                * OTHERWISE = ERROR VALUE
                *
                * DISK ATTRIBUTES:
                *
0EC5 001A       TRKSIZ EQU 26        26 SECTORS/TRACK
                *
0EC5 CE 000C R  SEMPTY LDX #SYSFCB   POINT TO SYSTEM FCB
0EC8 A7 09           STA A FCBDRV,X SET DRIVE NO.
                     TXAB
 0ECA 3F              SWI
 0ECB 02              FCB 2
0ECC CE 0036 R       LDX #BUFFER    SET BUFFER ADDRESS
                     XABX
 0ECF 3F              SWI
 0ED0 04              FCB 4
0ED1 A7 07           STA A FCBDBA,X
0ED3 E7 08           STA B FCBDBA+1,X
0ED5 6F 05           CLR FCBSTA,X   INIT. STATUS
                     OPEND          OPEN DIRECTORY
 0ED7 3F              SWI
 0ED8 17              FCB 23
                *
0ED9 A6 05      SEMPT2 LDA A FCBSTA,X CHECK STATUS
0EDB 27 07           BEQ SEMPT3     STATUS O.K.
                *
0EDD 81 01           CMP A #1       END-DIRECTORY?
0EDF 27 13           BEQ SEMPT4     YES
                *
0EE1 7E 0D21 R       JMP DIRERR     OTHERWISE ERROR
                *
0EE4 EE 27      SEMPT3 LDX FCBIND,X
0EE6 A6 00           LDA A 0,X      CHECK FIRST CHAR. OF SLOT
0EE8 81 20           CMP A #$20     BLANK?
0EEA 26 01           BNE *+3        NO
                *
0EEC 39              RTS            YES, FOUND EMPTY SLOT
                *
0EED CE 000C R       LDX #SYSFCB    POINT TO FCB
                     GETDR          GET NEXT DIR. BLOCK
 0EF0 3F              SWI
 0EF1 1A              FCB 26
0EF2 20 E5           BRA SEMPT2     KEEP LOOKING FOR EMPTY
                *
0EF4 A6 0B      SEMPT4 LDA A FCBSCT,X
0EF6 81 1A           CMP A #TRKSIZ END OF TRACK?
0EF8 26 01           BNE *+3        NO, FOUND EMPTY
                *
0EFA 39              RTS            YES, OUT OF SPACE
                *
0EFB 6F 05           CLR FCBSTA,X   RETURN GOOD STATUS
0EFD 39              RTS
                *
                * SEARCH DIRECTORY FOR UNAMBIGUOUS FILE REFERENCE
                * PASS IN INDEX REGISTER THE ADDRESS OF AN FCB
                * CONTAINING DESIRED FILE NAME AND DRIVE NO.
                * RETURNS ADDRESS OF DIRECTORY BLOCK IN FCBIND
                * RETURNS STATUS IN FCBSTA
                *       0=FOUND FILE
                *       1=FILE NOT FOUND
                * OTHERWISE=ERROR CODE
                *
0EFE            SFILE  PSHX          SAVE FCB ADDRESS
 0EFE 3F              SWI
 0EFF 05              FCB 5
0F00 A6 09           LDA A FCBDRV,X GET DRIVE NO.
0F02 CE 000C R       LDX #SYSFCB
0F05 A7 09           STA A FCBDRV,X PUT INTO SYSTEM FCB
                     TXAB
 0F07 3F              SWI
 0F08 02              FCB 2
0F09 CE 0036 R       LDX #BUFFER    PROVIDE A BUFFER ADDRESS
                     XABX
 0F0C 3F              SWI
 0F0D 04              FCB 4
0F0E A7 07           STA A FCBDBA,X
0F10 E7 08           STA B FCBDBA+1,X
0F12 6F 05           CLR FCBSTA,X   INIT. STATUS
                     OPEND          OPEN THE DIRECTORY ON DRIVE
 0F14 3F              SWI
 0F15 17              FCB 23
                *
0F16 A6 05      SFILE2 LDA A FCBSTA,X CHECK STATUS
0F18 27 0B           BEQ SFILE3     STATUS OK?
                *
0F1A 81 01           CMP A #1       END OF DIRECTORY?
0F1C 27 3B           BEQ SFILE5     YES
                *
                     PULX
 0F1E 3F              SWI
 0F1F 06              FCB 6
0F20 A7 05           STA A FCBSTA,X NO, ERROR STATUS
0F22 7E 0D21 R       JMP DIRERR     ISSUE ERROR MESSAGE
                *
0F25 EE 27      SFILE3 LDX FCBIND,X
0F27 A6 00           LDA A 0,X      CHECK DIRECTORY BLOCK
0F29 81 20           CMP A #$20     FIRST CHAR. OF NAME=BLANK?
0F2B 26 07           BNE SFILE4
                *
0F2D CE 000C R  SFNEXT LDX #SYSFCB   POINT TO SYSTEM FCB
                     GETDR          GET NEXT DIRECTORY BLOCK
```

```
2277 +  0F30 3F              SWI
2278 +  0F31 1A              FCB 26
2279    0F32 20 E2           BRA SFILE2    KEEP SEARCHING
2280                 *
2281                 SFILE4 PSHX          STACK DIRECTORY NAME ADDRESS
2282 +  0F34 3F              SWI
2283 +  0F35 05              FCB 5
2284    0F36 30              TSX
2285    0F37 EE 02           LDX 2,X       POINT TO SAVED FCB ADDRESS
2286    0F39 86 10           LDA A #FCBNAM
2287                         ADDAX         POINT TO NAME FIELD IN FCB
2288 +  0F3B 3F              SWI
2289 +  0F3C 09              FCB 9
2290                         PSHX          STACK SEARCH NAME ADDRESS
2291 +  0F3D 3F              SWI
2292 +  0F3E 05              FCB 5
2293    0F3F C6 0C           LDA B #12     12 CHARACTER COMPARISON
2294                         CMPC          COMPARE NAMES
2295 +  0F41 3F              SWI
2296 +  0F42 12              FCB 18
2297    0F43 31              INS           CLEAN STACK
2298    0F44 31              INS
2299    0F45 31              INS
2300    0F46 31              INS
2301    0F47 26 E4           BNE SFNEXT    NO MATCH, KEEP LOOKING
2302                 *
2303    0F49 CE 000C R SFOUND LDX #SYSFCB
2304    0F4C EE 27           LDX FCBIND,X  RECOVER DIR. BLOCK ADDRESS
2305                         TXAB
2306 +  0F4E 3F              SWI
2307 +  0F4F 02              FCB 2
2308                         PULX          RECOVER FCB ADDRESS
2309 +  0F50 3F              SWI
2310 +  0F51 06              FCB 6
2311    0F52 A7 27           STA A FCBIND,X
2312    0F54 E7 28           STA B FCBIND+1,X
2313    0F56 6F 05           CLR FCBSTA,X  GOOD STATUS
2314    0F58 39              RTS           RETURN TO CLI
2315                 *
2316                 SFILE5 PULX          RECOVER 'X'
2317 +  0F59 3F              SWI
2318 +  0F5A 06              FCB 6
2319    0F5B A7 05           STA A FCBSTA,X RETURN STATUS
2320    0F5D 39              RTS
2321                 *
2323                 * DELETE A FILE FROM DISK
2324                 * INDEX (STACKED) POINTS TO FCB WITH FILENAME AND DRIVE NO.
2325                 * RETURN ERROR STATUS IN FCBSTA
2326                 * CHECK ACCESS CODE FOR PROTECTION ERRORS
2327                 * MUST HAVE ALL OUTPUT FILES ON THIS DISK CLOSED
2328                 *
2329    0F5E 30      @DELETE TSX
2330    0F5F EE 05           LDX UXH,X             POINT TO FCB
2331    0F61 8D 9B           BSR SFILE             SEARCH DIRECTORY
2332    0F63 30              TSX
2333    0F64 EE 05           LDX UXH,X
2334    0F66 6D 05           TST FCBSTA,X          FOUND FILE?
2335    0F68 27 01           BEQ DEL2              YES
2336                 *
2337    0F6A 39              RTS                   NO, QUIT
2338                 *
2339    0F6B EE 27   DEL2    LDX FCBIND,X          POINT TO DIR. ENTRY
2340    0F6D 6D 0E           TST FIBACS,X          ACCESS CODE=0?
2341    0F6F 27 24           BEQ DEL3              O.K.
2342                 *
2343    0F71 CE 0F7E R       LDX #DELERR           NO, CANNOT DELETE
2344                         PRTMSG
2345 +  0F74 3F              SWI
2346 +  0F75 31              FCB 49
2347    0F76 30              TSX
2348    0F77 EE 05           LDX UXH,X             POINT TO FCB
2349    0F79 86 12           LDA A #18             ERROR CODE
2350    0F7B A7 05           STA A FCBSTA,X
2351    0F7D 39              RTS
2352                 *
2353    0F7E 20      DELERR FCC ' FILE DELETE-PROTECTED'
2354    0F94 0D              FCB $0D
2355                 *
2356    0F95 DE 29   DEL3    LDX FCBCHN            SEARCH OPEN FCBS
2357    0F97 27 46           BEQ DEL33
2358                 *
2359    0F99 6D 06   DEL30   TST FCBDTT,X          FCB OUTPUT?
2360    0F9B 27 0D           BEQ DEL31             NO, KEEP LOOKING
2361                 *
2362    0F9D A6 09           LDA A FCBDRV,X        GET DRIVE NO.
2363                         PSHX                  SAVE FCB POINTER
2364 +  0F9F 3F              SWI
2365 +  0FA0 05              FCB 5
2366    0FA1 30              TSX
2367    0FA2 EE 07           LDX UXH+2,X           POINT TO THIS FCB
2368    0FA4 A1 09           CMP A FCBDRV,X        SAME DRIVE?
2369    0FA6 27 08           BEQ DEL32             YES
2370                 *
2371                         PULX                  RESTORE POINTER
2372 +  0FA8 3F              SWI
2373 +  0FA9 06              FCB 6
2374    0FAA EE 25   DEL31   LDX FCBNFB,X          GET NEXT FCB
2375    0FAC 26 EB           BNE DEL30             KEEP LOOKING
2376                 *
2377    0FAE 20 2F           BRA DEL33             DONE
2378                 *
2379    0FB0 31      DEL32   INS                   CLEAN STACK
2380    0FB1 31              INS
2381    0FB2 CE 0FBF R       LDX #FOPERR           ERROR MESSAGE
2382                         PRTMSG
2383 +  0FB5 3F              SWI
2384 +  0FB6 31              FCB 49
2385    0FB7 86 12           LDA A #18             ERROR CODE
2386    0FB9 30              TSX
2387    0FBA EE 05           LDX UXH,X
2388    0FBC A7 05           STA A FCBSTA,X        RETURN ERROR CODE
2389    0FBE 39              RTS
2390                 *
2391    0FBF 20      FOPERR FCC ' DELETE ERROR-OPEN OUTPUT FILES'
2392    0FDE 0D              FCB $0D
2393                 *
2394    0FDF 30      DEL33   TSX
2395    0FE0 EE 05           LDX UXH,X             POINT TO FCB
2396    0FE2 EE 27           LDX FCBIND,X          POINT TO DIRECTORY ENTRY
2397    0FE4 A6 0F           LDA A FIBFTS,X
2398    0FE6 E6 10           LDA B FIBFTS+1,X      GET FIRST T/S OF FILE
2399    0FE8 30              TSX
```

```
2400   0FE9 EE 05          LDX UXH,X
2401   0FEB A7 1F          STA A FCBFTS,X          SAVE IN FCB
2402   0FED E7 20          STA B FCBFTS+1,X
2403   0FEF EE 27          LDX FCBIND,X            POINT TO DIRECTORY ENTRY
2404   0FF1 A6 11          LDA A FIBLTS,X
2405   0FF3 E6 12          LDA B FIBLTS+1,X        GET LAST T/S OF FILE
2406   0FF5 30             TSX
2407   0FF6 EE 05          LDX UXH,X
2408   0FF8 A7 21          STA A FCBLTS,X          STORE IN FCB
2409   0FFA E7 22          STA B FCBLTS+1,X
2410   0FFC EE 27          LDX FCBIND,X            POINT TO DIRECTORY ENTRY
2411   0FFE 86 20          LDA A #$20
2412   1000 A7 00          STA A 0,X               PUT BLANK INTO NAME FIELD
2413   1002 CE 000C R      LDX #SYSFCB
2414   1005 86 FF          LDA A #$FF
2415   1007 A7 06          STA A FCBDTT,X          MAKE 'OUTPUT'
2416                       IOHDR                   WRITE UPDATED DIRECTORY
2417 +  1009 3F             SWI
2418 +  100A 13             FCB 19
2419   100B 6F 06          CLR FCBDTT,X            RESTORE 'INPUT' STATE
2420   100D 6D 05    DEL3A TST FCBSTA,X            CHECK STATUS
2421   100F 27 01          BEQ DEL4                GOOD STATUS?
2422                *
2423   1011 39             RTS                     IF NOT, QUIT
2424                *
2425   1012 A6 09    DEL4  LDA A FCBDRV,X          GET DRIVE NO.
2426   1014 CE 08AC R      LDX #SAVFCB             USE SYSTEM FCB
2427   1017 A7 09          STA A FCBDRV,X          SET DRIVE NO.
2428   1019 86 00          LDA A #0                GET FREE-SPACE SECTOR
2429   101B C6 03          LDA B #3
2430   101D A7 0A          STA A FCBTRK,X          TRACK=0
2431   101F E7 0B          STA B FCBSCT,X          SECTOR=3
2432   1021 6F 06          CLR FCBDTT,X            INPUT
2433                       IOHDR                   READ SECTOR
2434 +  1023 3F             SWI
2435 +  1024 13             FCB 19
2436   1025 A6 05          LDA A FCBSTA,X          ERRORS?
2437   1027 26 E4          BNE DEL3A               YES
2438                *
2439   1029 EE 07          LDX FCBDBA,X            POINT TO SECTOR BUFFER
2440   102B A6 7E          LDA A SECSIZ-2,X        GET T/S OF FREE-SECTOR
2441   102D E6 7F          LDA B SECSIZ-1,X
2442   102F B7 0BDD R      STA A SAVEX             SAVE THEM
2443   1032 F7 0BDE R      STA B SAVEX+1
2444   1035 30             TSX
2445   1036 EE 05          LDX UXH,X
2446   1038 A6 1F          LDA A FCBFTS,X          GET FIRST T/S OF FILE
2447   103A E6 20          LDA B FCBFTS+1,X
2448   103C CE 08AC R      LDX #SAVFCB             USE SYSTEM FCB
2449   103F EE 07          LDX FCBDBA,X            POINT TO SECTOR BUFFER
2450   1041 A7 7E          STA A SECSIZ-2,X        UPDATE LINKAGE
2451   1043 E7 7F          STA B SECSIZ-1,X
2452   1045 36             PSH A                   SAVE 'A'
2453   1046 CE 08AC R      LDX #SAVFCB             USE SYSTEM FCB
2454   1049 A6 09          LDA A FCBDRV,X          GET DRIVE NUMBER
2455   104B 48             ASL A                   TIMES 2
2456   104C CE 002B        LDX #FRETAB             ACCESS FREE-SPACE TABLE
2457                       ADDAX
2458 +  104F 3F             SWI
2459 +  1050 09             FCB 9
2460   1051 32             PUL A                   RESTORE 'A'
2461   1052 A7 00          STA A 0,X               UPDATE TABLE
2462   1054 E7 01          STA B 1,X
2463   1056 CE 08AC R      LDX #SAVFCB             USE SYSTEM FCB
2464   1059 63 06          COM FCBDTT,X            MAKE FCB 'OUTPUT'
2465                       IOHDR                   WRITE UPDATED FREE-SPACE SECTOR
2466 +  105B 3F             SWI
2467 +  105C 13             FCB 19
2468   105D 6F 06          CLR FCBDTT,X            MAKE FCB 'INPUT'
2469   105F A6 05          LDA A FCBSTA,X          ERRORS?
2470   1061 26 AA          BNE DEL3A               YES
2471                *
2472   1063 30             TSX
2473   1064 EE 05          LDX UXH,X               POINT TO USER FCB
2474   1066 A6 21          LDA A FCBLTS,X          GET LAST T/S OF FILE
2475   1068 E6 22          LDA B FCBLTS+1,X
2476   106A CE 08AC R      LDX #SAVFCB             USE SYSTEM FCB
2477   106D A7 0A          STA A FCBTRK,X
2478   106F E7 0B          STA B FCBSCT,X
2479                       IOHDR                   READ THAT SECTOR
2480 +  1071 3F             SWI
2481 +  1072 13             FCB 19
2482   1073 A6 05          LDA A FCBSTA,X          ERRORS?
2483   1075 26 96          BNE DEL3A               YES
2484                *
2485   1077 EE 07          LDX FCBDBA,X            POINT TO SECTOR BUFFER
2486   1079 B6 0BDD R      LDA A SAVEX             GET T/S OF OLD FREE-SPACE
2487   107C F6 0BDE R      LDA B SAVEX+1
2488   107F A7 00          STA A 0,X               UPDATE LINKAGES
2489   1081 E7 01          STA B 1,X
2490   1083 CE 08AC R      LDX #SAVFCB             USE SYSTEM FCB
2491   1086 63 06          COM FCBDTT,X            MAKE FCB 'OUTPUT'
2492                       IOHDR                   WRITE UPDATED SECTOR
2493 +  1088 3F             SWI
2494 +  1089 13             FCB 19
2495   108A 6F 06          CLR FCBDTT,X            MAKE FCB 'INPUT'
2496   108C 39             RTS
2497                *
2499                * ROUTINE TO PARSE AN UNAMBIGUOUS FILE NAME
2500                *
2501                * [DRIVE:] FILENAME.EXT
2502                *
2503                * ADDRESS OF TEXT STRING IN 'CUCHAR'
2504                * FCB ADDRESS PASSED IN INDEX REGISTER
2505                * ERRORS RETURN '21' IN STATUS
2506                *
2507   108D 30          @FMTFCB TSX
2508   108E EE 05                LDX UXH,X       POINT TO FCB
2509   1090 6F 05                CLR FCBSTA,X    NO ERRORS YET
2510   1092 6F 09                CLR FCBDRV,X    DEFAULT DRIVE=0
2511                             NXTOK           GET A TOKEN
2512 +  1094 3F                   SWI
2513 +  1095 2F                   FCB 47
2514   1096 D6 25                LDA B RC        CHECK RC
2515   1098 C1 03                CMP B #3        NUMBER?
2516   109A 26 29                BNE PARS2       NO
2517                *
2518   109C 7D 0027              TST VALUE       VALID DRIVE NO.?
2519   109F 26 0A                BNE PARS1       NO
2520                *
2521   10A1 96 28                LDA A VALUE+1   VALID DRIVE NO.?
2522   10A3 81 03                CMP A #3        (4 DRIVES)
```

```
2523   10A5 22 04            BHI PARS1      NOT VALID
2524                  *
2525   10A7 A7 09            STA A FCBDRV,X INIT. DRIVE
2526   10A9 20 0E            BRA PARS1A
2527                  *
2528   10AB 30       PARS1   TSX
2529   10AC EE 05            LDX UXH,X      POINT TO FCB
2530   10AE 86 15            LDA A #21
2531   10B0 A7 05            STA A FCBSTA,X  RETURN ERROR CODE 21
2532   10B2 7F 0027          CLR VALUE
2533   10B5 7F 0028          CLR VALUE+1    RETURN NO VALUE
2534   10B8 39               RTS
2535                  *
2536                  PARS1A NXTOK          GET A TOKEN FROM CLI
2537 +  10B9 3F                SWI
2538 +  10BA 2F                FCB 47
2539   10BB D6 25            LDA B RC       CHECK RC
2540   10BD C1 3A            CMP B #':      COLON?
2541   10BF 26 EA            BNE PARS1      NO, ERROR
2542                  *
2543                         NXTOK          GET A TOKEN FROM CLI
2544 +  10C1 3F                SWI
2545 +  10C2 2F                FCB 47
2546   10C3 D6 25            LDA B RC       CHECK RC
2547   10C5 C1 01    PARS2   CMP B #1       UNAMBIG. NAME?
2548   10C7 27 08            BEQ PARS4      YES
2549                  *
2550   10C9 30       PARS3   TSX
2551   10CA EE 05            LDX UXH,X      POINT TO FCB
2552   10CC 86 15            LDA A #21
2553   10CE A7 05            STA A FCBSTA,X  RETURN ERROR STATUS 21
2554   10D0 39               RTS
2555                  *
2556   10D1 DE 20    PARS4   LDX DESCRA     POINT TO NAME
2557   10D3 FF 0BDD R        STX SAVEX      SAVE POINTER
2558   10D6 96 22            LDA A DESCRC   GET LENGTH
2559   10D8 B7 0BE4 R        STA A SAVEA    SAVE IT
2560                         NXTOK          GET A TOKEN FROM CLI
2561 +  10DB 3F                SWI
2562 +  10DC 2F                FCB 47
2563   10DD D6 25            LDA B RC       CHECK RC
2564   10DF C1 2E            CMP B #'.      PERIOD?
2565   10E1 26 E6            BNE PARS3      NO, ERROR
2566                  *
2567   10E3 7C 0BE4 R        INC SAVEA      COUNT PERIOD
2568                         NXTOK          GET A TOKEN FROM CLI
2569 +  10E6 3F                SWI
2570 +  10E7 2F                FCB 47
2571   10E8 D6 25            LDA B RC       CHECK RC
2572   10EA C1 01            CMP B #1       UNAMBIG. NAME?
2573   10EC 26 DB            BNE PARS3      NO, ERROR
2574                  *
2575   10EE D6 22            LDA B DESCRC   GET LENGTH OF EXT
2576   10F0 FB 0BE4 R        ADD B SAVEA    TOTAL LENGTH
2577   10F3 30               TSX
2578   10F4 EE 05            LDX UXH,X      POINT TO FCB
2579   10F6 86 10            LDA A #FCBNAM
2580                         ADDAX          POINT TO NAME FIELD IN FCB
2581 +  10F8 3F                SWI
2582 +  10F9 09                FCB 9
2583                         PSHX
2584 +  10FA 3F                SWI
2585 +  10FB 05                FCB 5
2586   10FC FE 0BDD R        LDX SAVEX      POINT TO NAME IN CLI
2587                         PSHX
2588 +  10FF 3F                SWI
2589 +  1100 05                FCB 5
2590                  FMTS    FORMAT NAME INTO FCB
2591 +  1101 3F                SWI
2592 +  1102 34                FCB 52
2593   1103 31               INS
2594   1104 31               INS            CLEAN STACK
2595   1105 31               INS
2596   1106 31               INS
2597   1107 5D               TST B          ERRORS?
2598   1108 26 BF            BNE PARS3      YES
2599                  *
2600   110A 39               RTS
2601                  *
2602                         END
```

```
@CHAIN 0E48 RN
@CLDST 0000 RN
@DELET 0F5E RN
@FMTFC 108D RN
@GTCMD 03FE RN
@LOADB 0966 RN
@NXTOK 0471 RN
@PRTER 03D4 RN
@PRTMS 0397 RN
@WRMST 0003 RN
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
ALLFIL 0BBA R
ASNCMD 062F R
ASNLIN 0696 R
BANNER 0300 R
BASEQU 2A2A M
BMEM   0033
BOOTCD 0639 R
BOOTLN 06B0 R
BS     0039
BUFFER 0036 R
BUFLIN 0BAD R
CHAIN  243A M
CHANER 0E7F R
CHANLN 0EA5 R
CHANME 0EB7 R
CHRTAB 0208 R
CLASS  0026
CLI    0000 RN
CLI0   033B R
CLI1   0347 R
CLI2   034E R
CLOSE  2369 M
CMDSRA 0355 R
CMDSRB 0389 R
CMDTAB 01B7 R
CMEM   0037
CMPC   231B M
CMWC   2572 M
COLDST 02A3 R
CONBUF 0248 R
CONRCB 0298 R
CRLF   02FE R
CUCHAR 0023
CVBTD  0CE6 R
CVDB   05BB RN
CVDB1  05D2 R
CVDB2  05D6 R
CVDEC1 0CEC R
CVDEC2 0CEF R
CVDEC5 0CFA R
CVHB   0564 RN
CVHB1  056F R
CVHBD  05A7 R
CVHBS  05AB R
DCOUNT 05B8 R
DEL2   0F6B R
DEL3   0F95 R
DEL30  0F99 R
DEL31  0FAA R
DEL32  0FB0 R
DEL33  0FDF R
DEL3A  100D R
DEL4   1012 R
DELCMD 061B R
DELERR 0F7E R
DELETE 2420 M
DELLIN 0666 R
DERNUM 03D0 R
DESCRA 0020
DESCRC 0022
DEVERR 03C4 R
DEVNAM 03C5 R
DIR1   0C34 R
DIR1A  0C3A R
DIR2   0C4A R
DIR3   0C52 R
DIR4   0C58 R
DIR5   0C6A R
DIR6   0C7B R
DIR7   0CAE R
DIRCD0 0C02 R
DIRCD1 0C20 R
DIRCD3 0C94 R
DIRCD4 0CD5 R
DIRCD5 0D4A R
DIRCMD 0BE5 R
DIRDRV 0B56 R
DIRERR 0D21 R
DIRFLD 0B59 R
DIRHDR 0B43 R
DIRLIN 0B82 R
DIRLN2 0BBD R
DIRLST 0D24 R
DIRNXT 0E31 R
DIV16  2524 M
DL     003A
DP     003B
DPCNT  003C
DSCAN  04CB R
DSCAN2 04E8 R
DUPERR 0AD3 R
DVAL   05B6 R
DX     0040
EJ     0041
EMEM   0035
ENDSCN 053E R
EQTAB  0006 RX
ES     0043
FCBACS 001E
FCBBAK 000E
FCBCHN 0029
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBDTT 0006
FCBEQT 0000
FCBFTS 001F
FCBFWD 000C
FCBGDT 0002
FCBIND 0027
FCBLTS 0021
FCBNAM 0010
FCBNFB 0025
FCBNMS 0023
FCBSCF 0029
FCBSCT 000B
FCBSTA 0005
FCBTRK 000A
FCBTYP 001D
FIBACS 000E
FIBDEF 2940 M
FIBFTS 000F
FIBLTS 0011
FIBNAM 0000
FIBNMS 0013
FIBTYP 000D
FMTFCB 2488 M
FMTS   2558 M
FOPERR 0FBF R
FORMAT 03AA R
FRETAB 002B
GCHRTB 0550 R
GCHRTR 0560 R
GETB2  09EC R
GETB3  09F0 R
GETBYT 09E0 R
GETDR  23EC M
GTCD2  046A R
GTCMD  24F0 M
HSCAN  0513 R
HSCAN1 051A R
HSCAN2 0537 R
HVAL   0562 R
INDEX  24BC M
IN1CD2 0B0D R
IN1CD3 0B31 R
IN1CMD 0AF9 R
IN1ERR 0B32 R
IN1TDK 253E M
IN1TLN 0B38 R
IOHDR  2335 M
JMPC2  0615 R
JMPCMD 0607 R
KIOK   0BD3 R
LDP    0044
LDPCNT 0045
LNKCMD 063E R
LNKLIN 06BB R
LOADB  246E M
LOADB2 097C R
LOADB3 09A2 R
LOADB4 09B5 R
LOADB5 09BC R
LOADB6 09CB R
LOD5   0965 R
LODCMD 0956 R
LPTFLG 0BE3 R
LWD    0046
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NL     003E
NSCAN  04F1 R
NSEC   0BAB R
NUMBER 03B7 R
NUMERR 0546 R
NXTO   0477 R
NXT1   048D R
NXT3   0493 R
NXT4   0496 R
NXT5   04A7 R
NXT6   04B0 R
NXT7   04BF R
NXTER  04C4 R
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
OUTHL  0E39 R
OUTHR  0E3D R
PARS1  10AB R
PARS1A 10B9 R
PARS2  10C5 R
PARS3  10C9 R
PARS4  10D1 R
PDTAB  0009 RX
PIPCMD 0620 R
PIPLIN 0673 R
PRTER2 03FD R
PRTERR 2454 M
PRTMSG 250A M
PS     0042
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTB2  08AB R
PUTBYT 08A1 R
PUTDR  2406 M
RC     0025
RCBDBA 0007
RCBDEF 258C M
RCBDTT 0006
RCBEQT 0000
RCBGDT 0002
RCBSTA 0005
READ   23B8 M
RENAM5 0A26 R
RENAM6 0A3C R
RENAM7 0A44 R
RENAM8 0A4A R
RENAM9 0ACD R
RENCMD 09F1 R
RENM5  0A9B R
RENMSG 0A2C R
REWIND 2384 M
SAV10  0881 R
SAV5   07D2 R
SAV5B  07D8 R
SAV6   082D R
SAV7   0836 R
SAV8   085A R
SAV9   086D R
SAVBUF 08D6 R
SAVCMD 07B4 R
SAVEA  0BE4 R
SAVERR 0899 R
SAVEX  0BDD R
SAVEX1 0BDF R
SAVFCB 08AC R
SECCMD 0625 R
SECERR 0AE3 R
SECLIN 067D R
SECSI7 0080
SECURE 0AE9 R
SEMPT2 0ED9 R
SEMPT3 0EE4 R
SEMPT4 0EF4 R
SEMPTY 0EC5 RN
SETCMD 062A R
SETLIN 068C R
SF1LE  0EFE RN
SF1LE2 0F16 R
SF1LE3 0F25 R
SF1LE4 0F34 R
SF1LE5 0F59 R
SFNEXT 0F2D R
SFOUND 0F49 R
STACK  01B6 R
STACMD 0634 R
STALIN 06A3 R
SUBABX 227F M
SUBAX  2299 M
SUBBUF 0734 R
SUBBX  22B3 M
SUBCMD 06C6 R
SUBEOF 043B R
SUBERR 06E7 R
SUBFCB 070A R
SUBFLG 0707 R
SUBLIN 06F4 R
SUBMT1 0425 R
SUBMT2 0445 R
SUBMT3 045B R
SUBMT4 0464 R
SUBTMP 0708 R
SUBXAB 2265 M
SYSFCB 000C RN
TABX   219C M
TB     003F
TENVL  05B9 R
TF1LE  0373 R
TMPX   0BE1 R
TRANS  0641 R
TRANS2 0665 R
TRKSI7 001A
TXAB   2183 M
TYPMSG 098F R
UA     0004
UB     0003
UXH    0005
UXL    0006
VALUE  0027
WARM1  0326 R
WARM2  032A R
WARM3  0330 R
WARMST 0320 R
WD     003D
WRITE  23D2 M
XABX   21B5 M
```

```
0001   0000 0000    N         NAM DIRECTORY
0002                 *
0003                 * OPEN, READ, WRITE DIRECTORY RECORDS ON DISK
0004                 * CP-68 AND ICOM 8 INCH FLOPPIES
0005                 *
0006                 * @OPEND OPENS DIRECTORY TO FIRST DATA BLOCK
0007                 * @GETDR GETS NEXT DATA BLOCK
0008                 * @PUTDR WRITES A DATA BLOCK
0009                 *
0010                 * ADDRESS OF FCB TO USE PASSED IN 'X' (ON STACK)
0011                 * MUST SET UP DRIVE NUMBER IN FCB
0012                 * MUST SET UP FCB AS 'DSK'
0013                 *
0014                 * RETURN STATUS IN FCBSTA:  0=BLOCK FOUND
0015                 *                           1=END OF DIRECTORY
0016                 *                           ELSE ERROR
0017                 *
0018                 * ADDRESS OF DATA BLOCK IN FCBIND
0019                 *
0020                 * FCB ADDRESS EQUATES
0021                 *
0022   0000 0005     FCBSTA EQU 5  STATUS FLAGS
0023   0000 0006     FCBDTT EQU 6  DIRECTION
0024   0000 0007     FCBDBA EQU 7  BUFFER ADDRESS
0025   0000 0009     FCBDRV EQU 9  DRIVE NO.
0026   0000 000A     FCBTRK EQU 10 TRACK NO.
0027   0000 000B     FCBSCT EQU 11 SECTOR NO.
0028   0000 0010     FCBNAM EQU 16 FILE-NAME FIELD
0029   0000 0027     FCBIND EQU 39 BUFFER INDEX
0030                 *
0031                 * REGISTER POINTERS
0032                 *
0033   0000 0005     UXH    EQU 5
0034   0000 0006     UXL    EQU 6
0035                 *
0036                 * DISK ATTRIBUTES
0037                 *
0038   0000 0080     SECSIZ EQU 128        128 BYTES/SECTOR
0039   0000 001A     TRKSIZ EQU 26         26 SECTORS/TRACK
0040   0000 0020     DIRBLK EQU 32         32 BYTES/DIRECTORY BLOCK
0041                 *
0042   0000 0003    N         ENT @OPEND
0043   0000 0034    N         ENT @GETDR
0044   0000 0064    N         ENT @PUTDR
0045                 *
0046   0000 7E 0000 X         EXT SYSFCB    SYSTEM FCB LOCATION
0047                 *
0048   0003 30       @OPEND TSX
0049   0004 EE 05             LDX UXH,X     POINT TO FCB
0050   0006 6F 06             CLR FCBDTT,X  INPUT
0051   0008 6F 0A             CLR FCBTRK,X  TRACK=0
0052   000A 86 04             LDA A #4      SECTOR=4 (START OF DIRECTORY)
0053                 *
0054   000C A7 0B    OPENDO STA A FCBSCT,X
0055   000E 6F 05             CLR FCBSTA,X  NO ERRORS
0056   0010 A6 07             LDA A FCBDBA,X
0057   0012 E6 08             LDA B FCBDBA+1,X      POINTER TO BUFFER START
0058   0014 A7 27             STA A FCBIND,X        INIT. DIR. BLOCK POINTER
0059   0016 E7 28             STA B FCBIND+1,X
0060                          IOHDR                 READ SECTOR
0061 +  0018 3F               SWI
0062 +  0019 13               FCB 19
0063   001A 4D                TST A         ERRORS?
0064   001B 26 0C             BNE OPEND2    YES
0065                 *
0066   001D EE 27    OPEND1 LDX FCBIND,X   POINT TO DATA BLOCK
0067   001F 6D 00             TST 0,X       FIRST CHAR=0?
0068   0021 27 09             BEQ OPEND3    YES, EOF
0069                 *
0070   0023 30                TSX
0071   0024 EE 05             LDX UXH,X     POINT TO FCB
0072   0026 6F 05             CLR FCBSTA,X  RETURN NO ERRORS
0073   0028 39                RTS           RETURN
0074                 *
0075   0029 A7 05    OPEND2 STA A FCBSTA,X       RETURN ERROR CODE
0076   002B 39                RTS
0077                 *
0078   002C 30       OPEND3 TSX
0079   002D EE 05             LDX UXH,X     POINT TO FCB
0080   002F 86 01             LDA A #1
0081   0031 A7 05             STA A FCBSTA,X       RETURN STATUS=1
0082   0033 39                RTS
0083                 *
0084                 *
0085                 *
0086   0034 30       @GETDR TSX
0087   0035 EE 05             LDX UXH,X     POINT TO FCB
0088   0037 A6 27             LDA A FCBIND,X
0089   0039 E6 28             LDA B FCBIND+1,X
0090   003B CB 20             ADD B #DIRBLK MOVE INDEX TO NEXT BLOCK
0091   003D 89 00             ADC A #0
0092   003F A7 27             STA A FCBIND,X
0093   0041 E7 28             STA B FCBIND+1,X
0094   0043 EE 07             LDX FCBDBA,X  BUFFER ADDRESS
0095   0045 36                PSH A         SAVE 'A'
0096   0046 86 80             LDA A #SECSIZ
0097                          ADDAX         BUFFER END ADDRESS
0098 +  0048 3F               SWI
0099 +  0049 09               FCB 9
0100   004A 32                PUL A         RESTORE 'A'
0101                          SUBXAB        COMPARE INDEX TO END-ADDRESS
0102 +  004B 3F               SWI
0103 +  004C 0B               FCB 11
0104   004D 27 08             BEQ GETDR2    NEED NEW SECTOR?
0105                 *
0106   004F 30                TSX           NO
0107   0050 EE 05             LDX UXH,X     POINT TO FCB
0108   0052 4F                CLR A
0109   0053 A7 05             STA A FCBSTA,X CLEAR ERROR STATUS
0110   0055 20 C6             BRA OPEND1    FINISH UP
0111                 *
0112   0057 30       GETDR2 TSX
0113   0058 EE 05             LDX UXH,X     POINT TO FCB
0114   005A A6 0B             LDA A FCBSCT,X
0115   005C 4C                INC A         NEXT SECTOR
0116   005D 81 1B             CMP A #TRKSIZ+1 END OF TRACK?
0117   005F 27 CB             BEQ OPEND3    YES, RETURN EOF
0118                 *
0119   0061 7E 000C R         JMP OPENDO    NO, GET NEW SECTOR
0120                 *
0121                 *
```

```
0122                    *
0123   0064 30          @PUTDR TSX
0124   0065 EE 05             LDX UXH,X       POINT TO FCB
0125   0067 EE 27             LDX FCBIND,X    GET ADDRESS OF DIR. BLOCK
0126                          PSHX            STACK IT
0127 +  0069 3F               SWI
0128 +  006A 05               FCB 5
0129   006B 30                TSX
0130   006C EE 07             LDX UXH+2,X     GET ADDRESS OF FCB
0131   006E 86 10             LDA A #FCBNAM
0132                          ADDAX           POINT TO NAME FIELD IN FCB
0133 +  0070 3F               SWI
0134 +  0071 09               FCB 9
0135                          PSHX            STACK IT
0136 +  0072 3F               SWI
0137 +  0073 05               FCB 5
0138   0074 C6 15             LDA B #21       21 BYTES TO MOVE
0139                          MOVC            MOVE FROM FCB TO DIR. BLOCK
0140 +  0076 3F               SWI
0141 +  0077 11               FCB 17
0142   0078 31                INS
0143   0079 31                INS             CLEAN STACK
0144   007A 31                INS
0145   007B 31                INS
0146   007C FE 0001 R         LDX SYSFCB+1    POINT TO SYSTEM FCB
0147   007F 63 06             COM FCBDT1,X    MAKE OUTPUT
0148                          IOHDR           ISSUE I/O REQUEST
0149 +  0081 3F               SWI
0150 +  0082 13               FCB 19
0151   0083 30                TSX
0152   0084 EE 05             LDX UXH,X       POINT TO FCB
0153   0086 A7 05             STA A FCBSTA,X RETURN STATUS
0154   0088 39                RTS
0155                    *
0156                          END
```

```
@GETDR 0034 RN
@OPEND 0003 RN
@PUTDR 0064 RN
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
CHAIN  243A M
CLOSE  2369 M
CMPC   231B M
CMWC   2572 M
DELETE 2420 M
DIRBLK 0020
DIRECT 0000 RN
DIV16  2524 M
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBDTT 0006
FCBIND 0027
FCBNAM 0010
FCBSCT 000B
FCBSTA 0005
FCBTRK 000A
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
GETDR  23EC M
GETDR2 0057 R
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
IOHDR  2335 M
LOADB  246E M
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
OPENDO 000C R
OPEND1 001D R
OPEND2 0029 R
OPEND3 002C R
PRTERR 2454 M
PRTMSG 250A M
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
SECSIZ 0080
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
SYSFCB 0000 RX
TABX   219C M
TRKSIZ 001A
TXAB   2183 M
UXH    0005
UXL    0006
WRITE  23D2 M
XABX   21B5 M
```

```
0001   0000 0000    N        NAM SFIO
0002                 *
0003                 * SEQUENTIAL FILE I/O PACKAGE
0004                 * @OPEN       OPEN SEQUENTIAL FILE FOR R/W
0005                 * @CLOSE      CLOSE SEQUENTIAL FILE
0006                 * @READ       READ A BYTE FROM SEQUENTIAL FILE
0007                 * @WRITE      WRITE A BYTE INTO A SEQUENTIAL FILE
0008                 * @REWD       REWIND A SEQUENTIAL FILE
0009                 *
0010                 * INDEX REGISTER (STACKED) POINTS TO FCB
0011                 * CHARACTERS PASSED IN 'A' REGISTER
0012                 *
0013                 * STATUS CODES:  (IN FCBSTA)
0014                 *
0015                 *      0=GOOD
0016                 *      1=END OF DIRECTORY
0017                 *      2=FILE IN USE
0018                 *      3=FILE ALREADY EXISTS
0019                 *      4=NO SUCH FILE
0020                 *      5=I/O ERROR
0021                 *      6=TOO MANY FILES FOR DIRECTORY
0022                 *      7=DISK FULL
0023                 *      8=END-FILE FOUND
0024                 *      9=BAD SECTOR
0025                 *     10=DEVICE NOT READY
0026                 *     13=ILLEGAL USE OF FCB
0027                 *     18=ILLEGAL OPERATION (WRITE TO READ FILE, ETC.)
0028                 *     21=BAD FILE NAME
0029                 *
0030                 * REGISTER POINTERS:
0031                 *
0032   0000 0003     UB       EQU 3
0033   0000 0004     UA       EQU 4
0034   0000 0005     UXH      EQU 5
0035   0000 0006     UXL      EQU 6
0036                 *
0037                 * BLOCK ADDRESSING POINTERS:
0038                 *
0039                        RCBDEF
0040 +   0000 0000      RCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0041 +   0000 0002      RCBGDT EQU 2          GENERIC DEVICE TYPE
0042 +   0000 0005      RCBSTA EQU 5          STATUS
0043 +   0000 0006      RCBDTT EQU 6          DATA TRANSFER TYPE
0044 +   0000 0007      RCBDBA EQU 7          DATA BUFFER ADDRESS
0045                        FCBDEF
0046 +   0000 0000      FCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0047 +   0000 0002      FCBGDT EQU 2          GENERIC DEVICE TYPE
0048 +   0000 0005      FCBSTA EQU 5          STATUS
0049 +   0000 0006      FCBDTT EQU 6          DATA TRANSFER TYPE
0050 +   0000 0007      FCBDBA EQU 7          DATA BUFFER ADDRESS
0051 +   0000 0009      FCBDRV EQU 9          DRIVE NUMBER
0052 +   0000 000A      FCBTRK EQU 10         TRACK NUMBER
0053 +   0000 000B      FCBSCT EQU 11         SECTOR NUMBER
0054 +   0000 000C      FCBFWD EQU 12         FWD LINK TRACK/SECTOR
0055 +   0000 000E      FCBBAK EQU 14         BACK LINK TRACK/SECTOR
0056 +   0000 0010      FCBNAM EQU 16         FILE NAME (8.3+EOT=13))
0057 +   0000 001D      FCBTYP EQU 29         FILE TYPE
0058 +   0000 001E      FCBACS EQU 30         FILE ACCESS CODE
0059 +   0000 001F      FCBFTS EQU 31         FIRST TRACK/SECTOR
0060 +   0000 0021      FCBLTS EQU 33         LAST TRACK/SECTOR
0061 +   0000 0023      FCBNMS EQU 35         NUMBER OF SECTORS
0062 +   0000 0025      FCBNFB EQU 37         NEXT FCB IN ACTIVE CHAIN
0063 +   0000 0027      FCBIND EQU 39         INDEX INTO DATA BUFFER
0064 +   0000 0029      FCBSCF EQU 41         SPACE COMPRESSION FLAG
0065                        FIBDEF
0066 +   0000 0000      FIBNAM EQU 0          FILE NAME (8.3 + EOT=13)
0067 +   0000 000D      FIBTYP EQU 13         FILE TYPE
0068 +   0000 000E      FIBACS EQU 14         FILE ACCESS CODE
0069 +   0000 000F      FIBFTS EQU 15         FIRST TRACK/SECTOR
0070 +   0000 0011      FIBLTS EQU 17         LAST TRACK/SECTOR
0071 +   0000 0013      FIBNMS EQU 19         NUMBER OF SECTORS
0072                 *
0073                 * DISK ATTRIBUTES:
0074                 *
0075   0000 0080     SECSIZ EQU 128        128 BYTES/SECTOR
0076   0000 0080     BUFSIZ FDB SECSIZ
0077                 *
0078                 *
0079   0002 000B    N        ENT @OPEN
0080   0002 019C    N        ENT @CLOSE
0081   0002 0251    N        ENT @READ
0082   0002 02EC    N        ENT @WRITE
0083   0002 03ED    N        ENT @REWD
0084                 *
0085   0002 7E 0000 X        EXT SEMPTY
0086   0005 7E 0000 X        EXT SFILE
0087   0008 7E 0000 X        EXT SYSFCB
0088                 *
0089                 * BASE PAGE POINTERS
0090                 *
0091   000B 0029     FCBCHN EQU $29        ACTIVE-FCB-CHAIN HEAD LINK
0092   000B 002B     FRETAB EQU $2B        FREE-SPACE POINTERS (4 DRIVES)
0094   000B 30       @OPEN  TSX
0095   000C EE 05            LDX UXH,X              POINT TO FCB
0096                         TXAB
0097 +   000E 3F              SWI
0098 +   000F 02              FCB 2
0099   0010 DE 29            LDX FCBCHN             START OF ACTIVE-FCB CHAIN
0100   0012 27 15            BEQ OPEN3              EMPTY CHAIN?
0101                 *
0102                 OPEN1  PSHX
0103 +   0014 3F              SWI
0104 +   0015 05              FCB 5
0105                         SUBABX                 FOUND FCB?
0106 +   0016 3F              SWI
0107 +   0017 0C              FCB 12
0108                         PULX
0109 +   0018 3F              SWI
0110 +   0019 06              FCB 6
0111   001A 27 06            BEQ OPEN2              YES, ERROR
0112                 *
0113   001C EE 25            LDX FCBNFB,X           GET NEXT CHAINED FCB
0114   001E 26 F4            BNE OPEN1              IF NOT END OF CHAIN, LOOP
0115                 *
0116   0020 20 07            BRA OPEN3              IF END, OK
0117                 *
0118                 OPEN2  TABX                   POINT TO FCB
0119 +   0022 3F              SWI
0120 +   0023 03              FCB 3
0121   0024 86 0D            LDA A #13              ERROR STATUS (FILE ALREADY OPEN
0122   0026 A7 05            STA A FCBSTA,X
```

```
0123   0028 39                RTS
0124                  *
0125                  OPEN3   TABX                    POINT TO FCB
0126 +  0029 3F                SWI
0127 +  002A 03                FCB 3
0128   002B 6D 06              TST FCBDTT,X           READ OR WRITE?
0129   002D 27 03              BEQ OPENR              READ
0130                  *
0131   002F 7E 00C4 R          JMP OPENW              WRITE
0132                  *
0133                  * OPEN SEQUENTIAL FILE FOR INPUT
0134                  *
0135   0032 BD 0005 R OPENR   JSR SFILE              SEARCH DIRECTORY
0136   0035 6D 05              TST FCBSTA,X           CHECK STATUS
0137   0037 27 05              BEQ OPENR1             GOOD?
0138                  *
0139   0039 86 04              LDA A #4               ERROR STATUS (NO SUCH FILE)
0140   003B A7 05              STA A FCBSTA,X
0141   003D 39                 RTS
0142                  *
0143   003E 86 1D     OPENR1  LDA A #FCBTYP
0144                          ADDAX                  POINT TO TYPE FIELD IN FCB
0145 +  0040 3F                SWI
0146 +  0041 09                FCB 9
0147                          PSHX                   STACK ADDRESS
0148 +  0042 3F                SWI
0149 +  0043 05                FCB 5
0150   0044 30                 TSX
0151   0045 EE 07              LDX UXH+2,X            POINT TO FCB
0152   0047 EE 27              LDX FCBIND,X           POINT TO DIRECTORY BLOCK
0153   0049 86 0D              LDA A #FIBTYP
0154                          ADDAX                  POINT TO TYPE FIELD IN DIR. BLOCK
0155 +  004B 3F                SWI
0156 +  004C 09                FCB 9
0157                          PSHX                   STACK ADDRESS
0158 +  004D 3F                SWI
0159 +  004E 05                FCB 5
0160   004F C6 08              LDA B #8               8 BYTES TO MOVE FROM DIR. TO FCB.
0161                          MOVC
0162 +  0051 3F                SWI
0163 +  0052 11                FCB 17
0164   0053 31                 INS
0165   0054 31                 INS                    CLEAN STACK
0166   0055 31                 INS
0167   0056 31                 INS
0168   0057 30                 TSX
0169   0058 EE 05              LDX UXH,X              POINT TO FCB
0170   005A A6 1F              LDA A FCBFTS,X
0171   005C E6 20              LDA B FCBFTS+1,X
0172   005E A7 0A              STA A FCBTRK,X         INIT. TRACK/SECTOR
0173   0060 E7 0B              STA B FCBSCT,X
0174                          IOHDR                  READ FIRST SECTOR OF FILE
0175 +  0062 3F                SWI
0176 +  0063 13                FCB 19
0177   0064 4D                 TST A                  ERROR?
0178   0065 27 03              BEQ OPENR2             NO
0179                  *
0180   0067 A7 05              STA A FCBSTA,X         RETURN ERROR STATUS
0181   0069 39                 RTS
0182                  *
0183   006A EE 07     OPENR2  LDX FCBDBA,X           POINT TO SECTOR BUFFER
0184   006C A6 00              LDA A 0,X              GET FORWARD LINKS
0185   006E E6 01              LDA B 1,X
0186   0070 30                 TSX
0187   0071 EE 05              LDX UXH,X              POINT TO FCB
0188   0073 A7 0C              STA A FCBFWD,X         PUT IN LINKS
0189   0075 E7 0D              STA B FCBFWD+1,X
0190   0077 EE 07              LDX FCBDBA,X           POINT TO SECTOR BUFFER
0191   0079 A6 02              LDA A 2,X              GET BACKWARD LINKS
0192   007B E6 03              LDA B 3,X
0193   007D 30                 TSX
0194   007E EE 05              LDX UXH,X              POINT TO FCB
0195   0080 A7 0E              STA A FCBBAK,X         PUT IN BACKWARD LINKS
0196   0082 E7 0F              STA B FCBBAK+1,X
0197   0084 A6 07     OPENR3  LDA A FCBDBA,X
0198   0086 E6 08              LDA B FCBDBA+1,X
0199   0088 CB 04              ADD B #4               INIT. BUFFER INDEX
0200   008A 89 00              ADC A #0
0201   008C A7 27              STA A FCBIND,X
0202   008E E7 28              STA B FCBIND+1,X
0203                  *
0204                  * PUT FCB ONTO ACTIVE-FCB CHAIN
0205                  * COMMON TO READ AND WRITE
0206                  * (X) POINTS TO FCB
0207                  *
0208   0090 6F 25     OPEN4   CLR FCBNFB,X           MAKE FCB END OF CHAIN
0209   0092 6F 26              CLR FCBNFB+1,X
0210   0094 DE 29              LDX FCBCHN             SEARCH CHAIN FOR END LINK
0211   0096 26 0E              BNE OPEN5              EMPTY CHAIN?
0212                  *
0213   0098 30                 TSX
0214   0099 A6 05              LDA A UXH,X            GET FCB ADDRESS INTO (A,B)
0215   009B E6 06              LDA B UXL,X
0216   009D 97 29              STA A FCBCHN           INIT. CHAIN
0217   009F D7 2A              STA B FCBCHN+1
0218                          TABX                   RESTORE FCB ADDRESS
0219 +  00A1 3F                SWI
0220 +  00A2 03                FCB 3
0221   00A3 6F 05              CLR FCBSTA,X           GOOD STATUS
0222   00A5 39                 RTS
0223                  *
0224   00A6 6D 25     OPEN5   TST FCBNFB,X           AT END OF CHAIN?
0225   00A8 26 16              BNE OPEN6              NO
0226                  *
0227   00AA 6D 26              TST FCBNFB+1,X         AT END OF CHAIN?
0228   00AC 26 12              BNE OPEN6              NO
0229                  *
0230                          PSHX                   SAVE END OF CHAIN ADDRESS
0231 +  00AE 3F                SWI
0232 +  00AF 05                FCB 5
0233   00B0 30                 TSX
0234   00B1 A6 07              LDA A UXH+2,X          GET FCB ADDRESS INTO (A,B)
0235   00B3 E6 08              LDA B UXL+2,X
0236                          PULX
0237 +  00B5 3F                SWI
0238 +  00B6 06                FCB 6
0239   00B7 A7 25              STA A FCBNFB,X         PATCH CHAIN
0240   00B9 E7 26              STA B FCBNFB+1,X
0241                          TABX                   RESTORE FCB ADDRESS
0242 +  00BB 3F                SWI
0243 +  00BC 03                FCB 3
0244   00BD 6F 05              CLR FCBSTA,X           GOOD STATUS
```

```
00BF 39                 RTS
                *
00C0 EE 25      OPEN6   LDX FCBNFB,X        GET NEXT LINK IN CHAIN
00C2 20 E2              BRA OPEN5
                *
                *
                * OPEN SEQUENTIAL FILE FOR OUTPUT
                *
00C4 BD 0005 R  OPENW   JSR SFILE           SEARCH DIRECTORY
00C7 A6 05              LDA A FCBSTA,X      CHECK STATUS
00C9 81 01              CMP A #1            FILE FOUND?
00CB 27 05              BEQ OPENW1          NO
                *
00CD 86 03              LDA A #3            ERROR STATUS (FILE EXISTS)
00CF A7 05              STA A FCBSTA,X
00D1 39                 RTS
                *
00D2 A6 09      OPENW1  LDA A FCBDRV,X      GET DRIVE NO.
00D4 BD 0002 R          JSR SEMPTY          SEARCH FOR DIR. SPACE
00D7 FE 0009 R          LDX SYSFCB+1        POINT TO SYSTEM FCB
00DA A6 05              LDA A FCBSTA,X      CHECK STATUS
00DC 27 0E              BEQ OPENW4          GOOD?
                *
00DE 81 01              CMP A #1            NO ROOM IN DIRECTORY?
00E0 27 06              BEQ OPENW3          YES
                *
00E2 30         OPENW2  TSX
00E3 EE 05              LDX UXH,X           POINT TO FCB
00E5 A7 05              STA A FCBSTA,X      RETURN ERROR STATUS
00E7 39                 RTS
                *
00E8 86 06      OPENW3  LDA A #6            ERROR STATUS (NO ROOM)
00EA 20 F6              BRA OPENW2
                *
00EC A6 27      OPENW4  LDA A FCBIND,X      GET DIR. BLOCK ADDRESS FROM SYS.
00EE E6 28              LDA B FCBIND+1,X
00F0 30                 TSX
00F1 EE 05              LDX UXH,X           POINT TO FCB
00F3 A7 27              STA A FCBIND,X      SAVE ADDRESS
00F5 E7 28              STA B FCBIND+1,X
00F7 6F 23              CLR FCBNMS,X        INIT. NO. SECTORS=0
00F9 6F 24              CLR FCBNMS+1,X
00FB 6F 21              CLR FCBLTS,X        INIT. LAST T/S=0,0
00FD 6F 22              CLR FCBLTS+1,X
00FF 6F 0E              CLR FCBBAK,X        INIT. BACKWARD POINTERS
0101 6F 0F              CLR FCBBAK+1,X
0103 A6 09              LDA A FCBDRV,X      GET DRIVE NO.
0105 84 03              AND A #$03          LIMIT RANGE (0-3)
0107 48                 ASL A               2 BYTES PER ENTRY
0108 CE 002B            LDX #FRETAB         ACCESS FREE-SPACE TABLE
                        ADDAX
+ 010B 3F                SWI
+ 010C 09                FCB 9
                        PSHX                SAVE TABLE POINTER
+ 010D 3F                SWI
+ 010E 05                FCB 5
010F A6 00              LDA A 0,X           GET FREE T/S
0111 E6 01              LDA B 1,X
0113 4D                 TST A               TABLE INIT. YET?
0114 27 03              BEQ OPNW4           NO
                *
0116 5D                 TST B               TABLE INIT. YET?
0117 26 27              BNE OPENW6          YES
                *
                * MUST READ IN FREE-SPACE SECTOR AND INIT. FREE-SPACE TABLE
                *
0119 30         OPNW4   TSX
011A EE 07              LDX UXH+2,X         POINT TO FCB
011C 6F 06              CLR FCBDTT,X        MAKE 'INPUT'
011E 86 00              LDA A #0            TRACK=0
0120 C6 03              LDA B #3            SECTOR=3 (FREE-SPACE RECORD)
0122 A7 0A              STA A FCBTRK,X
0124 E7 0B              STA B FCBSCT,X
                        IOHDR               ISSUE READ COMMAND
+ 0126 3F                SWI
+ 0127 13                FCB 19
0128 63 06              COM FCBDTT,X        PUT 'OUPUT' BACK
012A 4D                 TST A               ERROR?
012B 27 05              BEQ OPENW5          NO
                *
012D A7 05              STA A FCBSTA,X      RETURN ERROR CODE
012F 31                 INS                 CLEAN STACK
0130 31                 INS
0131 39                 RTS                 QUIT
                *
0132 EE 07      OPENW5  LDX FCBDBA,X        POINT TO DATA BUFFER
0134 A6 7E              LDA A SECSIZ-2,X    GET T/S OF FREE SPACE
0136 E6 7F              LDA B SECSIZ-1,X
                        PULX                RECOVER FREE-SPACE TABLE POINTE
+ 0138 3F                SWI
+ 0139 06                FCB 6
                        PSHX                PUT BACK ON STACK
+ 013A 3F                SWI
+ 013B 05                FCB 5
013C A7 00              STA A 0,X           INIT. FREE-SPACE TABLE
013E E7 01              STA B 1,X
                *
0140 30         OPENW6  TSX
0141 EE 07              LDX UXH+2,X         POINT TO FCB
0143 4D                 TST A               AT END OF DISK?
0144 27 03              BEQ OPNW6A          YES
                *
0146 5D                 TST B               AT END OF DISK?
0147 26 07              BNE OPNW6B          NO
                *
0149 86 07      OPNW6A  LDA A #7            END OF DISK ERROR
014B A7 05              STA A FCBSTA,X
014D 31                 INS
014E 31                 INS
014F 39                 RTS
                *
0150 A7 1F      OPNW6B  STA A FCBFTS,X      INIT. FIRST T/S
0152 E7 20              STA B FCBFTS+1,X
0154 A7 0A              STA A FCBTRK,X      POINT TO THAT T/S
0156 E7 0B              STA B FCBSCT,X
                        PUTDR               UPDATE DIRECTORY
+ 0158 3F                SWI
+ 0159 1B                FCB 27
015A 6D 05              TST FCBSTA,X        CHECK STATUS
015C 27 03              BEQ OPENW7          GOOD
                *
015E 31                 INS
```

```
0367   015F 31              INS                 CLEAN STACK
0368   0160 39              RTS                 QUIT
0369                  *
0370   0161 6F 06     OPENW7 CLR FCBDTT,X        MAKE 'INPUT'
0371                         IOHDR               ISSUE READ COMMAND
0372 +  0163 3F              SWI
0373 +  0164 13              FCB 19
0374   0165 63 06            COM FCBDTT,X        RESTORE 'OUTPUT'
0375   0167 4D               TST A               CHECK FOR ERROR
0376   0168 27 05            BEQ OPENW8          GOOD
0377                  *
0378   016A A7 05            STA A FCBSTA,X      RETURN ERROR CODE
0379   016C 31               INS
0380   016D 31               INS                 CLEAN STACK
0381   016E 39               RTS                 QUIT
0382                  *
0383   016F EE 07     OPENW8 LDX FCBDBA,X        POINT TO DATA BUFFER
0384   0171 A6 00            LDA A 0,X           GET FORWARD POINTERS
0385   0173 E6 01            LDA B 1,X
0386                         PULX                RECOVER FREE-SPACE TABLE POINTER
0387 +  0175 3F              SWI
0388 +  0176 06              FCB 6
0389   0177 4D               TST A               OUT OF SPACE?
0390   0178 26 0B            BNE OPENW9          NO
0391                  *
0392   017A 5D               TST B               OUT OF SPACE?
0393   017B 26 08            BNE OPENW9          NO
0394                  *
0395   017D 86 07            LDA A #7            RETURN ERROR CODE (OUT OF SPACE)
0396   017F 30               TSX
0397   0180 EE 05            LDX UXH,X           POINT TO FCB
0398   0182 A7 05            STA A FCBSTA,X
0399   0184 39               RTS
0400                  *
0401   0185 A7 00     OPENW9 STA A 0,X           UPDATE FREE-SPACE TABLE
0402   0187 E7 01            STA B 1,X
0403   0189 30               TSX
0404   018A EE 05            LDX UXH,X           POINT TO FCB
0405   018C EE 07            LDX FCBDBA,X        POINT TO DATA BUFFER
0406   018E C6 7C            LDA B #SECSIZ-4
0407   0190 6F 04     OPNW9A CLR 4,X             CLEAR OUT BUFFER
0408   0192 08               INX
0409   0193 5A               DEC B
0410   0194 26 FA            BNE OPNW9A
0411                  *
0412   0196 30               TSX
0413   0197 EE 05            LDX UXH,X           POINT TO FCB
0414   0199 7E 0084 R        JMP OPENR3          FINISH UP LIKE READ
0416   019C 30        @CLOSE TSX
0417   019D A6 05            LDA A UXH,X
0418   019F E6 06            LDA B UXL,X    GET FCB ADDRESS
0419   01A1 DE 29            LDX FCBCHN     GET HEAD OF FCB CHAIN
0420   01A3 27 2E            BEQ NOCHN      NO ACTIVE FCBS?
0421                  *
0422                         PSHX           SAVE X
0423 +  01A5 3F              SWI
0424 +  01A6 05              FCB 5
0425                         SUBABX         AT DESIRED FCB?
0426 +  01A7 3F              SWI
0427 +  01A8 0C              FCB 12
0428                         PULX           RESTORE X
0429 +  01A9 3F              SWI
0430 +  01AA 06              FCB 6
0431   01AB 26 0A            BNE NOTFND     NO
0432                  *
0433   01AD A6 25            LDA A FCBNFB,X
0434   01AF E6 26            LDA B FCBNFB+1,X
0435   01B1 97 29            STA A FCBCHN   NEW CHAIN HEAD
0436   01B3 D7 2A            STA B FCBCHN+1
0437   01B5 20 23            BRA CLOSE2     FINISH
0438                  *
0439   01B7 A1 25     NOTFND CMP A FCBNFB,X  AT DESIRED FCB?
0440   01B9 26 14            BNE NXTFCB     NO
0441                  *
0442   01BB E1 26            CMP B FCBNFB+1,X AT DESIRED FCB?
0443   01BD 26 10            BNE NXTFCB     NO
0444                  *
0445                  * FIX ACTIVE FCB CHAIN TO GO AROUND THIS FCB
0446                  * (X) POINTS TO PREVIOUS FCB
0447                  * (A,B) POINTS TO THIS FCB
0448                  *
0449                         PSHX
0450 +  01BF 3F              SWI
0451 +  01C0 05              FCB 5
0452                         TABX           POINT TO THIS FCB
0453 +  01C1 3F              SWI
0454 +  01C2 03              FCB 3
0455   01C3 A6 25            LDA A FCBNFB,X GET ITS LINKAGE
0456   01C5 E6 26            LDA B FCBNFB+1,X
0457                         PULX           POINT TO PREVIOUS FCB
0458 +  01C7 3F              SWI
0459 +  01C8 06              FCB 6
0460   01C9 A7 25            STA A FCBNFB,X UPDATE ITS LINKAGE
0461   01CB E7 26            STA B FCBNFB+1,X
0462   01CD 20 0B            BRA CLOSE2     FINISH PROCESSING
0463                  *
0464   01CF EE 25     NXTFCB LDX FCBNFB,X   GET NEXT FCB ADDRESS
0465   01D1 26 E4            BNE NOTFND     IF NOT END OF CHAIN, KEEP GOING
0466                  *
0467                  NOCHN  TABX           POINT TO THIS FCB
0468 +  01D3 3F              SWI
0469 +  01D4 03              FCB 3
0470   01D5 86 0D            LDA A #13      RETURN ERROR CODE (CAN'T FIND FCB)
0471   01D7 A7 05            STA A FCBSTA,X
0472   01D9 39               RTS
0473                  *
0474                  *
0475                  *
0476   01DA 30        CLOSE2 TSX
0477   01DB EE 05            LDX UXH,X      POINT TO THIS FCB
0478   01DD 6D 06            TST FCBDTT,X   READ OR WRITE?
0479   01DF 26 01            BNE CLOSEW     WRITE
0480                  *
0481   01E1 39               RTS            READ IS DONE
0482                  *
0483   01E2 6D 0A     CLOSEW TST FCBTRK,X   AT END OF DISK?
0484   01E4 27 04            BEQ CLSW1      YES
0485                  *
0486   01E6 6D 0B            TST FCBSCT,X   AT END OF DISK?
0487   01E8 26 0A            BNE CLSW2      NO
0488                  *
0489   01EA A6 0E     CLSW1  LDA A FCBBAK,X
```

```
0490   01EC E6 0F           LDA B FCBBAK+1,X
0491   01EE A7 0A           STA A FCBTRK,X
0492   01F0 E7 0B           STA B FCBSCT,X
0493   01F2 20 0E           BRA CLSW3      ERROR FIX-UP FOR END-OF-DISK
0494                *
0495                CLSW2   IOHDR          WRITE OUT LAST SECTOR OF FILE
0496 +  01F4 3F              SWI
0497 +  01F5 13              FCB 19
0498   01F6 A6 23           LDA A FCBNMS,X
0499   01F8 E6 24           LDA B FCBNMS+1,X
0500   01FA CB 01           ADD B #1       ONE MORE SECTOR IN COUNT
0501   01FC 89 00           ADC A #0
0502   01FE A7 23           STA A FCBNMS,X
0503   0200 E7 24           STA B FCBNMS+1,X
0504                *
0505                * LAST SECTOR NOW ON DISK
0506                * UPDATE DIRECTORY INFORMATION
0507                *
0508   0202 6F 06   CLSW3   CLR FCBDTT,X   MAKE 'INPUT'
0509   0204 BD 0005 R       JSR SFILE      FIND DIRECTORY SLOT
0510   0207 30              TSX
0511   0208 EE 05           LDX UXH,X      POINT TO FCB
0512   020A 63 06           COM FCBDTT,X   RESTORE 'OUTPUT'
0513   020C 6D 05           TST FCBSTA,X   CHECK STATUS
0514   020E 27 01           BEQ CLOSE3     GOOD
0515                *
0516   0210 39              RTS            IF NO GOOD, PUNT!!!
0517                *
0518   0211 A6 0A   CLOSE3 LDA A FCBTRK,X   GET LAST TRACK WRITTEN
0519   0213 E6 0B           LDA B FCBSCT,X           SECTOR
0520   0215 A7 21           STA A FCBLTS,X  PUT INTO FCB POSITION
0521   0217 E7 22           STA B FCBLTS+1,X
0522                        PUTDR          WRITE DATA INTO DIRECTORY
0523 +  0219 3F              SWI
0524 +  021A 1B              FCB 27
0525   021B 6D 05           TST FCBSTA,X   GOOD WRITE?
0526   021D 27 01           BEQ CLOSE4     YES
0527                *
0528   021F 39              RTS            NO GOOD, QUIT!!
0529                *
0530   0220 6F 06   CLOSE4 CLR FCBDTT,X     MAKE 'INPUT'
0531   0222 86 00           LDA A #0       TRACK=0
0532   0224 C6 03           LDA B #3       SECTOR=3  (FREE-SPACE RECORD)
0533   0226 A7 0A           STA A FCBTRK,X
0534   0228 E7 0B           STA B FCBSCT,X
0535                        IOHDR          READ FREE-SPACE RECORD
0536 +  022A 3F              SWI
0537 +  022B 13              FCB 19
0538   022C 63 06           COM FCBDTT,X   RESTORE 'OUTPUT'
0539   022E 4D              TST A          ERROR?
0540   022F 27 01           BEQ CLOSE5     NO
0541                *
0542   0231 39              RTS            BAD READ, QUIT!!
0543                *
0544   0232 A6 09   CLOSE5 LDA A FCBDRV,X   GET DRIVE NO.
0545   0234 84 03           AND A #$03     LIMIT RANGE (0-3)
0546   0236 48              ASL A          2 BYTES/ENTRY
0547   0237 CE 002B         LDX #FRETAB    ACCESS FREE-SPACE TABLE
0548                        ADDAX
0549 +  023A 3F              SWI
0550 +  023B 09              FCB 9
0551   023C A6 00           LDA A 0,X      GET TRACK/SECTOR
0552   023E E6 01           LDA B 1,X
0553   0240 30              TSX
0554   0241 EE 05           LDX UXH,X      POINT TO FCB
0555   0243 EE 07           LDX FCBDBA,X   POINT TO DATA BUFFER
0556   0245 A7 7E           STA A SECSIZ-2,X  PUT NEW T/S INTO BUFFER
0557   0247 E7 7F           STA B SECSIZ-1,X
0558   0249 30              TSX
0559   024A EE 05           LDX UXH,X      POINT TO FCB
0560                        IOHDR          WRITE OUT UPDATED SECTOR
0561 +  024C 3F              SWI
0562 +  024D 13              FCB 19
0563   024E A7 05           STA A FCBSTA,X  SAVE STATUS
0564   0250 39              RTS
0566   0251 30      @READ   TSX
0567   0252 EE 05           LDX UXH,X      POINT TO FCB
0568   0254 6D 06           TST FCBDTT,X   INPUT REQUESTED?
0569   0256 27 05           BEQ READ2      OK
0570                *
0571   0258 86 12           LDA A #18      ERROR CODE
0572   025A A7 05           STA A FCBSTA,X
0573   025C 39              RTS            QUIT
0574                *
0575   025D A6 27   READ2  LDA A FCBIND,X   CHECK FOR END OF BUFFER
0576   025F E6 28           LDA B FCBIND+1,X
0577   0261 E0 08           SUB B FCBDBA+1,X
0578   0263 A2 07           SBC A FCBDBA,X
0579   0265 B1 0000 R       CMP A BUFSIZ   AT END?
0580   0268 26 05           BNE READ2A     NO
0581                *
0582   026A F1 0001 R       CMP B BUFSIZ+1  AT END?
0583   026D 27 36           BEQ READ3      YES
0584                *
0585   026F EE 27   READ2A LDX FCBIND,X   POINT TO BUFFER
0586   0271 A6 00           LDA A 0,X      GET BYTE
0587                *
0588                * CHECK FOR SPACE COMPRESSION
0589                * IN SPACE-COMPRESSION MODE, BYTE= NEGATIVE SPACE COUNT
0590                *
0591   0273 2A 1E           BPL READ2C     NOT A COMPRESSED SPACE
0592                *
0593   0275 30              TSX
0594   0276 EE 05           LDX UXH,X      POINT TO FCB
0595   0278 6D 29           TST FCBSCF,X   IN COMPRESSED MODE?
0596   027A 27 15           BEQ READ2B     NO
0597                *
0598   027C EE 27           LDX FCBIND,X   POINT TO BUFFER
0599   027E 4C              INC A          ONE FEWER SPACE
0600   027F 26 04           BNE NOTLST     LAST SPACE?
0601                *
0602   0281 86 20           LDA A #$20     IF SO, REPLACE WITH SPACE
0603   0283 20 0E           BRA READ2C
0604                *
0605   0285 A7 00   NOTLST STA A 0,X       PUT NEW CHAR. IN BUFFER
0606   0287 86 20           LDA A #$20     OUTPUT A SPACE
0607   0289 30              TSX
0608   028A A7 04           STA A UA,X
0609   028C EE 05           LDX UXH,X      POINT TO FCB
0610   028E 6F 05           CLR FCBSTA,X   GOOD STATUS
0611   0290 39              RTS            DONE
0612                *
```

```
0613                            *
0614                            *
0615   0291 EE 27       READ2B  LDX FCBIND,X  POINT TO BUFFER
0616                            *
0617   0293 08          READ2C  INX           MOVE BUFFER POINTER
0618                            PSHX          STACK IT
0619 +  0294 3F                  SWI
0620 +  0295 05                  FCB 5
0621   0296 30                  TSX
0622   0297 A7 06               STA A UA+2,X  RETURN BYTE
0623   0299 32                  PUL A         GET POINTER FROM STACK
0624   029A 33                  PUL B
0625   029B 30                  TSX
0626   029C EE 05               LDX UXH,X     POINT TO FCB
0627   029E A7 27               STA A FCBIND,X SAVE NEW POINTER
0628   02A0 E7 28               STA B FCBIND+1,X
0629   02A2 6F 05               CLR FCBSTA,X  GOOD STATUS
0630   02A4 39                  RTS           DONE
0631                            *
0632                            *
0633                            *
0634   02A5 A6 0A       READ3   LDA A FCBTRK,X
0635   02A7 A1 21               CMP A FCBLTS,X AT END OF FILE?
0636   02A9 26 0B               BNE READ4      NO
0637                            *
0638   02AB A6 0B               LDA A FCBSCT,X
0639   02AD A1 22               CMP A FCBLTS+1,X AT END OF FILE?
0640   02AF 26 05               BNE READ4        NO
0641                            *
0642   02B1 86 08               LDA A #8         RETURN END-FILE STATUS
0643   02B3 A7 05       READ3A  STA A FCBSTA,X
0644   02B5 39                  RTS
0645                            *
0646   02B6 A6 0C       READ4   LDA A FCBFWD,X   GET FORWARD LINK T/S
0647   02B8 E6 0D               LDA B FCBFWD+1,X
0648   02BA A7 0A               STA A FCBTRK,X   PUT LINK INTO T/S
0649   02BC E7 0B               STA B FCBSCT,X
0650                            IOHDR            READ LINKED SECTOR
0651 +  02BE 3F                  SWI
0652 +  02BF 13                  FCB 19
0653   02C0 4D                  TST A            ERROR CHECK
0654   02C1 26 F0               BNE READ3A       RETURN ERROR CODE
0655                            *
0656   02C3 A6 07               LDA A FCBDBA,X
0657   02C5 E6 08               LDA B FCBDBA+1,X
0658   02C7 CB 04               ADD B #4         RE-INIT.  INDEX
0659   02C9 89 00               ADC A #0         SKIP FOUR BYTES OF LINK
0660   02CB A7 27               STA A FCBIND,X
0661   02CD E7 28               STA B FCBIND+1,X
0662   02CF EE 07               LDX FCBDBA,X     POINT TO BUFFER
0663   02D1 A6 00               LDA A 0,X        GET NEW FORWARD LINKS
0664   02D3 E6 01               LDA B 1,X
0665   02D5 30                  TSX
0666   02D6 EE 05               LDX UXH,X        POINT TO FCB
0667   02D8 A7 0C               STA A FCBFWD,X   STORE FORWARD LINKS
0668   02DA E7 0D               STA B FCBFWD+1,X
0669   02DC EE 07               LDX FCBDBA,X     POINT TO BUFFER
0670   02DE A6 02               LDA A 2,X        GET NEW BACKWARD LINKS
0671   02E0 E6 03               LDA B 3,X
0672   02E2 30                  TSX
0673   02E3 EE 05               LDX UXH,X        POINT TO FCB
0674   02E5 A7 0E               STA A FCBBAK,X   STORE BACKWARD LINKS
0675   02E7 E7 0F               STA B FCBBAK+1,X
0676   02E9 7E 026F R           JMP READ2A       NOW READ BYTE
0677                            *
0679   02EC 30          @WRITE  TSX
0680   02ED EE 05               LDX UXH,X      POINT TO FCB
0681   02EF 6D 06               TST FCBDTT,X   CHECK FOR OUTPUT
0682   02F1 26 05               BNE WRITE2     OK
0683                            *
0684   02F3 86 12               LDA A #18
0685   02F5 A7 05               STA A FCBSTA,X RETURN ERROR CODE
0686   02F7 39                  RTS            QUIT
0687                            *
0688   02F8 A6 27       WRITE2  LDA A FCBIND,X CHECK FOR END OF BUFFER
0689   02FA E6 28               LDA B FCBIND+1,X
0690   02FC E0 08       WRIT20  SUB B FCBDBA+1,X
0691   02FE A2 07               SBC A FCBDBA,X
0692   0300 B1 0000 R           CMP A BUFSIZ   END OF BUFFER?
0693   0303 26 05               BNE WRIT2A     NO
0694                            *
0695   0305 F1 0001 R           CMP B BUFSIZ+1 END OF BUFFER?
0696   0308 27 45               BEQ WRITE3     YES
0697                            *
0698   030A 30          WRIT2A  TSX
0699   030B A6 04               LDA A UA,X     GET CHARACTER TO BE WRITTEN
0700   030D EE 05               LDX UXH,X      POINT TO FCB
0701   030F 6D 29               TST FCBSCF,X   IN SPACE-COMPRESSION MODE?
0702   0311 27 27               BEQ WRIT2B     NO
0703                            *
0704                            *
0705                            * HANDLE COMPRESSION OF SPACES HERE
0706                            * USE NEGATIVE COUNT OF SPACES
0707                            *
0708   0313 84 7F               AND A #$7F     STRIP HIGH BIT
0709   0315 81 20               CMP A #$20     SPACE?
0710   0317 26 21               BNE WRIT2B     NO
0711                            *
0712   0319 EE 27               LDX FCBIND,X   POINT TO BUFFER
0713   031B E6 00               LDA B 0,X      GET BYTE FROM BUFFER
0714   031D 2A 0B               BPL NEWSPC     FIRST SPACE?
0715                            *
0716   031F 5A                  DEC B          ONE MORE SPACE ADDED
0717   0320 2A 0C               BPL SPC128     SPACE COUNT>128?
0718                            *
0719   0322 E7 00       STRSPC  STA B 0,X      PUT COUNT BACK
0720   0324 30                  TSX
0721   0325 EE 05               LDX UXH,X      POINT TO FCB
0722   0327 6F 05               CLR FCBSTA,X   GOOD STATUS
0723   0329 39                  RTS            DONE
0724                            *
0725   032A C6 FF       NEWSPC  LDA B #$FF     FIRST COMPRESSED SPACE
0726   032C 20 F4               BRA STRSPC
0727                            *
0728   032E 08          SPC128  INX            MOVE POINTER
0729                            TXAB
0730 +  032F 3F                  SWI
0731 +  0330 02                  FCB 2
0732   0331 30                  TSX
0733   0332 EE 05               LDX UXH,X      POINT TO FCB
0734   0334 A7 27               STA A FCBIND,X SAVE NEW POINTER
0735   0336 E7 28               STA B FCBIND+1,X
```

```
0736   0338 20 C2              BRA WRIT20    CONTINUE WITH SPACE
0737                    *
0738                    *
0739                    *
0740   033A EE 27       WRIT2B LDX FCBIND,X  POINT TO BUFFER
0741   033C 6D 00             TST 0,X       CHAR. ALREADY THERE?
0742   033E 26 EE             BNE SPC128    YES
0743                    *
0744   0340 A7 00             STA A 0,X     STORE CHARACTER IN BUFFER
0745   0342 08                INX           MOVE POINTER
0746                           TXAB
0747 +  0343 3F                SWI
0748 +  0344 02                FCB 2
0749   0345 30                 TSX
0750   0346 EE 05              LDX UXH,X     POINT TO FCB
0751   0348 A7 27              STA A FCBIND,X PUT NEW INDEX IN FCB
0752   034A E7 28              STA B FCBIND+1,X
0753   034C 6F 05              CLR FCBSTA,X  GOOD STATUS
0754   034E 39                 RTS           DONE
0755                    *
0756   034F A6 09       WRITE3 LDA A FCBDRV,X  GET DRIVE NO.
0757   0351 84 03             AND A #$03    LIMIT RANGE (0-3)
0758   0353 CE 002B           LDX #FRETAB   ACCESS FREE-SPACE TABLE
0759   0356 48                ASL A         TWO BYTES/ENTRY
0760                           ADDAX
0761 +  0357 3F                SWI
0762 +  0358 09                FCB 9
0763   0359 A6 00             LDA A 0,X     GET FREE-TRACK
0764   035B 27 18             BEQ WRIT3A    END OF DISK?
0765                    *
0766   035D E6 01             LDA B 1,X     GET FREE-SECTOR
0767   035F 27 14             BEQ WRIT3A    END OF DISK?
0768                    *
0769   0361 30                 TSX
0770   0362 EE 05              LDX UXH,X     POINT TO FCB
0771   0364 EE 07              LDX FCBDBA,X  POINT TO DATA BUFFER
0772   0366 A7 00              STA A 0,X     NEW FORWARD LINK TRACK
0773   0368 E7 01              STA B 1,X     NEW FORWARD LINK SECTOR
0774   036A 30                 TSX
0775   036B EE 05              LDX UXH,X     POINT TO FCB
0776                           IOHDR         WRITE OUT SECTOR
0777 +  036D 3F                SWI
0778 +  036E 13                FCB 19
0779   036F 4D                 TST A         ERROR?
0780   0370 27 06              BEQ WRITE4    NO
0781                    *
0782   0372 A7 05              STA A FCBSTA,X  RETURN ERROR STATUS
0783   0374 39                 RTS
0784                    *
0785   0375 7E 03E5 R   WRIT3A JMP WRITE7          MAKE ERROR RETURN
0786                    *
0787   0378 A6 23       WRITE4 LDA A FCBNMS,X     GET SECTOR COUNT
0788   037A E6 24              LDA B FCBNMS+1,X
0789   037C CB 01              ADD B #1          INCREMENT IT
0790   037E 89 00              ADC A #0
0791   0380 A7 23              STA A FCBNMS,X
0792   0382 E7 24              STA B FCBNMS+1,X
0793   0384 A6 0A              LDA A FCBTRK,X    GET TRACK JUST WRITTEN
0794   0386 E6 0B              LDA B FCBSCT,X    GET SECTOR
0795   0388 A7 0E              STA A FCBBAK,X    PUT IN BACK LINK
0796   038A E7 0F              STA B FCBBAK+1,X

0797   038C A6 09              LDA A FCBDRV,X    GET DRIVE NO.
0798   038E 84 03              AND A #$03        LIMIT RANGE (0-3)
0799   0390 CE 002B            LDX #FRETAB       ACCESS FREE-SPACE TABLE
0800   0393 48                 ASL A             2 BYTES/ENTRY
0801                           ADDAX
0802 +  0394 3F                 SWI
0803 +  0395 09                 FCB 9
0804   0396 A6 00              LDA A 0,X         GET NEXT TRACK
0805   0398 E6 01              LDA B 1,X         GET NEXT SECTOR
0806                           PSHX              SAVE INDEX TO FREE-SPACE TABLE
0807 +  039A 3F                 SWI
0808 +  039B 05                 FCB 5
0809   039C 30                 TSX
0810   039D EE 07              LDX UXH+2,X       POINT TO FCB
0811   039F A7 0A              STA A FCBTRK,X    NEW TRACK TO GET
0812   03A1 E7 0B              STA B FCBSCT,X    NEW SECTOR
0813   03A3 6F 06              CLR FCBDTT,X      MAKE INPUT
0814                           IOHDR             READ IN SECTOR
0815 +  03A5 3F                 SWI
0816 +  03A6 13                 FCB 19
0817   03A7 63 06              COM FCBDTT,X      REPLACE 'OUTPUT'
0818   03A9 4D                 TST A             ERROR?
0819   03AA 27 05              BEQ WRITE5        NO
0820                    *
0821   03AC A7 05              STA A FCBSTA,X    RETURN ERROR CODE
0822   03AE 31                 INS               CLEAN STACK
0823   03AF 31                 INS
0824   03B0 39                 RTS
0825                    *
0826   03B1 EE 07       WRITE5 LDX FCBDBA,X      POINT TO BUFFER
0827   03B3 A6 00              LDA A 0,X         GET NEW LINK TRACK
0828   03B5 E6 01              LDA B 1,X         GET NEW SECTOR
0829                           PULX              RECOVER FREE-SPACE INDEX
0830 +  03B7 3F                 SWI
0831 +  03B8 06                 FCB 6
0832   03B9 A7 00              STA A 0,X         PUT LINK INTO TABLE
0833   03BB E7 01              STA B 1,X
0834   03BD 30                 TSX
0835   03BE EE 05              LDX UXH,X         POINT TO FCB
0836   03C0 A7 0C              STA A FCBFWD,X    SET FORWARD LINKS
0837   03C2 E7 0D              STA B FCBFWD+1,X
0838   03C4 A6 07              LDA A FCBDBA,X    GET BUFFER ADDRESS
0839   03C6 E6 08              LDA B FCBDBA+1,X
0840   03C8 CB 04              ADD B #4          RE-INIT. BUFFER INDEX
0841   03CA 89 00              ADC A #0
0842   03CC A7 27              STA A FCBIND,X
0843   03CE E7 28              STA B FCBIND+1,X
0844   03D0 A6 0E              LDA A FCBBAK,X    GET BACKWARD LINK
0845   03D2 E6 0F              LDA B FCBBAK+1,X
0846   03D4 EE 07              LDX FCBDBA,X      POINT TO BUFFER
0847   03D6 A7 02              STA A 2,X         PUT IN BACKWARD LINKS
0848   03D8 E7 03              STA B 3,X
0849   03DA C6 7C              LDA B #SECSIZ-4
0850   03DC 6F 04       WRITE6 CLR 4,X           ZERO OUT REST OF BUFFER
0851   03DE 08                 INX
0852   03DF 5A                 DEC B
0853   03E0 26 FA              BNE WRITE6
0854                    *
0855   03E2 7E 030A R           JMP WRIT2A       CONTINUE WITH NEW SECTOR
0856                    *
0857   03E5 86 07       WRITE7 LDA A #7          DISK FULL ERROR
```

```
0858   03E7 30              TSX
0859   03E8 EE 05           LDX UXH,X              POINT TO FCB
0860   03EA A7 05           STA A FCBSTA,X
0861   03EC 39              RTS
0863   03ED 30        @REWD TSX
0864   03EE EE 05           LDX UXH,X              POINT TO FCB
0865   03F0 6D 06           TST FCBDTT,X           CHECK FOR INPUT
0866   03F2 27 05           BEQ REWD2              OK?
0867                *
0868   03F4 86 12           LDA A #18              ERROR CODE (REWIND OUTPUT FILE)
0869   03F6 A7 05           STA A FCBSTA,X
0870   03F8 39              RTS
0871                *
0872   03F9 6F 05     REWD2 CLR FCBSTA,X
0873                        CLOSE                  CLOSE FILE
0874 +  03FB 3F              SWI
0875 +  03FC 15              FCB 21
0876   03FD 6D 05           TST FCBSTA,X           CHECK STATUS
0877   03FF 27 01           BEQ REWD3              OK?
0878                *
0879   0401 39              RTS                    RETURN ERROR STATUS
0880                *
0881                REWD3 OPEN                     RE-OPEN FILE
0882 +  0402 3F              SWI
0883 +  0403 14              FCB 20
0884   0404 39              RTS
0885                *
0886                        END
```

```
@CLOSE 019C RN
@OPEN  000B RN
@READ  0251 RN
@REWD  03ED RN
@WRITE 02EC RN
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
BUFSIZ 0000 R
CHAIN  243A M
CLOSE  2369 M
CLOSE2 01DA R
CLOSE3 0211 R
CLOSE4 0220 R
CLOSE5 0232 R
CLOSEW 01E2 R
CLSW1  01EA R
CLSW2  01F4 R
CLSW3  0202 R
CMPC   231B M
CMWC   2572 M
DELETE 2420 M
DIV16  2524 M
FCBACS 001E
FCBBAK 000E
FCBCHN 0029
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBDTT 0006
FCBEQT 0000
FCBFTS 001F
FCBFWD 000C
FCBGDT 0002
FCBIND 0027
FCBLTS 0021
FCBNAM 0010
FCBNFB 0025
FCBNMS 0023
FCBSCF 0029
FCBSCT 000B
FCBSTA 0005
FCBTRK 000A
FCBTYP 001D
FIBACS 000E
FIBDEF 2940 M
FIBFTS 000F
FIBLTS 0011
FIBNAM 0000
FIBNMS 0013
FIBTYP 000D
FMTFCB 2488 M
FMTS   2558 M
FRETAB 002B
GETDR  23EC M
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
```

```
IOHDR  2335 M
LOADB  246E M
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NEWSPC 032A R
NOCHN  01D3 R
NOTFND 01B7 R
NOTLST 0285 R
NXTFCB 01CF R
NXTOK  24D6 M
OPEN   234F M
OPEN1  0014 R
OPEN2  0022 R
OPEN3  0029 R
OPEN4  0090 R
OPEN5  00A6 R
OPEN6  00C0 R
OPEND  239E M
OPENR  0032 R
OPENR1 003E R
OPENR2 006A R
OPENR3 0084 R
OPENW  00C4 R
OPENW1 00D2 R
OPENW2 00E2 R
OPENW3 00E8 R
OPENW4 00EC R
OPENW5 0132 R
OPENW6 0140 R
OPENW7 0161 R
OPENW8 016F R
OPENW9 0185 R
OPNW4  0119 R
OPNW6A 0149 R
OPNW6B 0150 R
OPNW9A 0190 R
PRTERR 2454 M
PRTMSG 250A M
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
RCBDBA 0007
RCBDEF 258C M
RCBDTT 0006
RCBEQT 0000
RCBGDT 0002
RCBSTA 0005
READ   23B8 M
READ2  025D R
READ2A 026F R
READ2B 0291 R
READ2C 0293 R
READ3  02A5 R
READ3A 02B3 R
READ4  02B6 R
REWD2  03F9 R
REWD3  0402 R
```

```
REWIND 2384 M
SECSIZ 0080
SEMPTY 0002 RX
SFILE  0005 RX
SFIO   0000 RN
SPC128 032E R
STKSPC 0322 R
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
SYSFCB 0008 RX
TABX   219C M
TXAB   2183 M
UA     0004
UB     0003
UXH    0005
UXL    0006
WRIT20 02FC R
WRIT2A 030A R
WRIT2B 033A R
WRIT3A 0375 R
WRITE  23D2 M
WRITE2 02F8 R
WRITE3 034F R
WRITE4 0378 R
WRITE5 03B1 R
WRITE6 03DC R
WRITE7 03E5 R
XABX   21B5 M
```

```
0001   0000 0000    N        NAM ICOMDRV
0002                 *
0003                 * DISK DRIVERS FOR ICOM
0004                 * SINGLE-SECTOR READ/WRITE
0005                 * TO BE USED WITH CP-68 SYSTEM
0006                 *
0007   0000 0000    N        ENT @INTDK     INITIALIZE INTERFACE
0008   0000 002C    N        ENT .RDSEC     READ A SECTOR
0009   0000 0097    N        ENT .WTSEC     WRITE A SECTOR
0010                 *
0011                 * PIA DEFINTIONS
0012                 *
0013   0000 EC00     INDAT  EQU $EC00       DATA/STATUS INPUT
0014   0000 EC01     INCTL  EQU $EC01       DATA/STATUS CONTROL
0015   0000 EC02     CMDDAT EQU $EC02       COMMAND OUTPUT
0016   0000 EC03     CMDCTL EQU $EC03       COMMAND CONTROL
0017   0000 EC06     OUTDAT EQU $EC06       DATA OUTPUT
0018   0000 EC07     OUTCTL EQU $EC07       DATA OUTPUT CONTROL
0019                 *
0020                 * CONTROL COMMAND DEFINTIONS
0021                 *
0022   0000 0002     READX  EQU $02         READ
0023   0000 0004     WRITEX EQU $04         WRITE
0024   0000 0006     RDCRC  EQU $06         READ CRC
0025   0000 0008     SEEK   EQU $08         SEEK
0026   0000 000A     CLRERF EQU $0A         RESET ERROR FLAGS
0027   0000 000C     SEEKTO EQU $0C         SEEK TRACK 0
0028   0000 0010     LDTRAD EQU $10         LOAD TRACK ADDRESS
0029   0000 0020     LDUS   EQU $20         LOAD UNIT/SECTOR
0030   0000 0030     LDWBF  EQU $30         LOAD WRITE BUFFER
0031   0000 0040     SHFTRB EQU $40         SHIFT READ BUFFER
0032   0000 0080     CLEAR  EQU $80         CLEAR
0033                 *
0034                 * FCB ADDRESS DEFINITIONS
0035                 *
0036   0000 0005     FCBSTA EQU 5           STATUS
0037   0000 0007     FCBDBA EQU 7           DATA BUFFER ADDRESS
0038   0000 0009     FCBDRV EQU 9           UNIT NUMBER
0039   0000 000A     FCBTRK EQU 10          TRACK NUMBER
0040   0000 000B     FCBSCT EQU 11          SECTOR NUMBER
0041                 *
0042   0000 0006     UA     EQU 6           RETURN 'A' REGISTER
0043   0000 0007     UXH    EQU 7           USER X-REG (RCBADR)
0044                 *
0045                 * NOTE: .RDSEC AND .WTSEC CALLED AS SUBROUTINES
0046                 *
0047                 * INITIALIZE DISK INTERFACE
0048                 *
0049   0000 7F EC01  @INTDK CLR INCTL       CLEAR CONTROL REGISTER
0050   0003 7F EC03         CLR CMDCTL
0051   0006 7F EC07         CLR OUTCTL
0052   0009 7F EC00         CLR INDAT       DDR=INPUT
0053                 *
0054   000C 86 FF           LDA A #$FF
0055   000E B7 EC02         STA A CMDDAT    DDR=OUTPUT
0056   0011 B7 EC06         STA A OUTDAT
0057                 *
0058   0014 86 04           LDA A #$04
0059   0016 B7 EC01         STA A INCTL     INIT. CREG
0060   0019 B7 EC07         STA A OUTCTL
0061                 *
0062   001C 86 2C           LDA A #$2C
0063   001E B7 EC03         STA A CMDCTL    INIT. CREG
0064                 *
0065   0021 86 80           LDA A #CLEAR
0066   0023 B7 EC02         STA A CMDDAT    ISSUE CLEAR COMMAND
0067                 *
0068   0026 86 0C           LDA A #SEEKTO
0069   0028 BD 00F3 R       JSR OUTCMD      SEEK TRACK 0
0070                 *
0071   002B 39              RTS
0072                 *
0073                 * SET UP FOR SINGLE-SECTOR READ
0074                 * GET DATA FROM FCB
0075                 * FCB ADDRESS IN (A,B)
0076                 *
0077                  RDSEC TABX             POINT X TO FCB
0078 +  002C 3F             SWI
0079 +  002D 03             FCB 3
0080   002E A6 09           LDA A FCBDRV,X
0081   0030 0C              CLC
0082   0031 46              ROR A           MOVE UNIT BITS
0083   0032 46              ROR A
0084   0033 46              ROR A
0085   0034 AA 0B           ORA A FCBSCT,X
0086   0036 E6 0A           LDA B FCBTRK,X
0087   0038 EE 07           LDX   FCBDBA,X
0088                 *
0089                 * READ A SECTOR INTO BUFFER
0090                 *
0091                 *       A=U/S
0092                 *       B=TRACK
0093                 *       X=BUFFER ADDRESS
0094                 *
0095   003A BD 00E7 R GETBUF JSR XMITUS.     SEND UNIT/SECTOR
0096   003D BD 011A R       JSR DRIVCK      DRIVE OK?
0097   0040 24 02           BCC GETBF0      YES
0098                 *
0099   0042 20 49           BRA QUIT        NO, DRIVE BAD
0100                 *
0101   0044 17       GETBF0 TBA             A=TRACK
0102   0045 BD 010C R       JSR SEEKTK      SEEK TRACK
0103                 *
0104   0048 C6 05           LDA B #5        5 RETRIES
0105                 *
0106   004A 86 02    GETBF1 LDA A #READX
0107   004C BD 00F3 R       JSR OUTCMD      ISSUE READ COMMAND
0108   004F B6 EC00         LDA A INDAT     GET STATUS
0109   0052 85 08           BIT A #$08      CRC ERROR?
0110   0054 27 0A           BEQ GETBF2      NO
0111                 *
0112   0056 BD 0104 R       JSR ERFRST      RESET ERROR FLAGS
0113   0059 5A              DEC B           TRIED ENOUGH?
0114   005A 26 EE           BNE GETBF1      NO, KEEP TRYING
0115                 *
0116   005C 86 05           LDA A #5        RETURN ERROR CODE=5
0117   005E 20 2D           BRA QUIT
0118                 *
0119   0060 85 80    GETBF2 BIT A #$80      DDAM?
0120   0062 27 04           BEQ GETBF3      NO
0121                 *
```

```
0122   0064 86 09          LDA A #9       RETURN ERROR CODE=9
0123   0066 20 25          BRA QUIT
0124                       *
0125   0068 C6 80    GETBF3 LDA B #128     128 BYTES IN SECTOR
0126                       *
0127   006A 86 3C    GETBF4 LDA A #$3C
0128   006C B7 EC03        STA A CMDCTL   INIT. COMMAND CONTROL REGISTER
0129   006F 86 40          LDA A #SHFTRB
0130   0071 B7 EC02        STA A CMDDAT   READ DATA COMMAND
0131                       *
0132   0074 B6 EC00        LDA A INDAT    GET A BYTE
0133   0077 36             PSH A          SAVE IT
0134                       *
0135   0078 86 2C          LDA A #$2C
0136   007A B7 EC03        STA A CMDCTL   RESET COMMAND CONTROL
0137                       *
0138   007D 86 40          LDA A #SHFTRB  STROBE
0139   007F B7 EC02        STA A CMDDAT   READ
0140   0082 7F EC02        CLR CMDDAT     BUFFER
0141                       *
0142   0085 32             PUL A          GET DATA BYTE
0143   0086 A7 00          STA A 0,X      MOVE TO BUFFER
0144   0088 08             INX
0145   0089 5A             DEC B          DONE WITH BUFFER?
0146   008A 26 DE          BNE GETBF4     NO
0147                       *
0148   008C 4F             CLR A          YES, SET RC
0149                       *
0150   008D 30       QUIT  TSX
0151   008E A7 06          STA A UA,X     RETURN 'A' CONTENTS
0152   0090 EE 07          LDX UXH,X      GET RCBADR
0153   0092 AA 05          ORA A FCBSTA,X
0154   0094 A7 05          STA A FCBSTA,X RETURN STATUS
0155   0096 39             RTS
0156                       *
0157                       * SET UP FOR SINGLE SECTOR WRITE
0158                       * ADDRESS OF FCB PASSED IN (A,B)
0159                       *
0160                        WTSEC TABX          POINT X TO FCB
0161 +  0097 3F                  SWI
0162 +  0098 03                  FCB 3
0163   0099 A6 09          LDA A FCBDRV,X
0164   009B 0C             CLC
0165   009C 46             ROR A
0166   009D 46             ROR A
0167   009E 46             ROR A          MOVE UNIT BITS
0168   009F AA 0B          ORA A FCBSCT,X
0169   00A1 E6 0A          LDA B FCBTRK,X
0170   00A3 EE 07          LDX   FCBDBA,X
0171                       *
0172                       * WRITE A SECTOR TO DISK
0173                       *
0174                       *      A=U/S
0175                       *      B=TRACK
0176                       *      X=BUFFER ADDRESS
0177                       *
0178   00A5 36       WRTBUF PSH A          SAVE U/S
0179   00A6 37              PSH B          SAVE TRACK
0180                       *
0181   00A7 C6 80          LDA B #128     128 BYTES IN BUFFER
0182                       *
0183   00A9 A6 00    WRTBF0 LDA A 0,X      GET A BYTE
0184   00AB 08              INX
0185   00AC B7 EC06        STA A OUTDAT   OUTPUT BYTE
0186   00AF 86 30          LDA A #LDWBF
0187   00B1 B7 EC02        STA A CMDDAT   LOAD WRITE BUFFER
0188   00B4 5A             DEC B
0189   00B5 26 F2          BNE WRTBF0     LOOP UNTIL DONE
0190                       *
0191   00B7 33             PUL B          RESTORE TRACK
0192   00B8 32             PUL A          RESTORE U/S
0193                       *
0194   00B9 BD 00E7 R      JSR XMITUS     SEND U/S
0195   00BC BD 011A R      JSR DRIVCK     DRIVE OK?
0196   00BF 24 02          BCC WRTBF1     YES
0197                       *
0198   00C1 20 CA          BRA QUIT       NO, DRIVE BAD
0199                       *
0200   00C3 17       WRTBF1 TBA            A=TRACK
0201   00C4 BD 010C R      JSR SEEKTK     SEEK TRACK
0202                       *
0203   00C7 C6 05          LDA B #5       5 RETRIES
0204                       *
0205   00C9 86 04    WRTBF2 LDA A #WRITEX
0206   00CB BD 00F3 R      JSR OUTCMD     SEND WRITE COMMAND
0207                       *
0208   00CE 86 06          LDA A #RDCRC
0209   00D0 BD 00F3 R      JSR OUTCMD     SEND CHECK CRC COMMAND
0210   00D3 B6 EC00        LDA A INDAT    GET STATUS
0211   00D6 85 08          BIT A #$08     OK?
0212   00D8 27 0A          BEQ WRTBF3     YES
0213                       *
0214   00DA BD 0104 R      JSR ERFRST     RESET ERROR FLAGS
0215   00DD 5A             DEC B
0216   00DE 26 E9          BNE WRTBF2     RETRIED 5 TIMES YET?
0217                       *
0218   00E0 86 05          LDA A #5       YES, ERROR CODE=5
0219   00E2 20 A9          BRA QUIT
0220                       *
0221   00E4 4F       WRTBF3 CLR A          SET RC
0222   00E5 20 A6          BRA QUIT
0223                       *
0224                       * TRANSMIT UNIT/SECTOR FROM 'A'
0225                       *
0226   00E7 BD 0104 R XMITUS JSR ERFRST     CLEAR ERROR FLAGS
0227   00EA B7 EC06        STA A OUTDAT   OUTPUT U/S
0228   00ED 86 20          LDA A #LDUS
0229   00EF B7 EC02        STA A CMDDAT   SEND LOAD U/S COMMAND
0230   00F2 39             RTS
0231                       *
0232                       * OUTPUT COMMAND FROM 'A'
0233                       *
0234   00F3 36       OUTCMD PSH A          SAVE COMMAND
0235   00F4 B6 EC00        LDA A INDAT    CLEAR BUSY FLAG
0236   00F7 32             PUL A          RESTORE COMMAND
0237   00F8 B7 EC02        STA A CMDDAT   OUTPUT COMMAND
0238                       *
0239   00FB B6 EC01 OUTCM1 LDA A INCTL    DONE?
0240   00FE 2A FB          BPL OUTCM1     WAIT FOR DONE
0241                       *
0242   0100 B6 EC00        LDA A INDAT    CLEAR BUSY
0243   0103 39             RTS
```

```
0244                    *
0245                    * CLEAR ERROR FLAGS
0246                    *
0247   0104 36          ERFRST PSH A          SAVE U/S
0248   0105 86 0A             LDA A #CLRERF
0249   0107 B7 EC02           STA A CMDDAT   ISSUE RESET-FLAGS COMMAND
0250   010A 32                PUL A          RESTORE U/S
0251   010B 39                RTS
0252                    *
0253                    * SEEK TRACK IN 'A'
0254                    *
0255   010C B7 EC06    SEEKTK STA A OUTDAT   OUTPUT TRACK
0256   010F 86 10             LDA A #LDTRAD
0257   0111 B7 EC02           STA A CMDDAT   SEND LOAD-TRACK-ADDRESS COMMAND
0258   0114 86 08             LDA A #SEEK
0259   0116 BD 00F3 R         JSR OUTCMD     SEND SEEK COMMAND
0260   0119 39                RTS
0261                    *
0262                    * DRIVE CHECK
0263                    *
0264   011A B6 EC00    DRIVCK LDA A INDAT    GET STATUS
0265   011D 84 20             AND A #$20     DISK READY?
0266   011F 26 03             BNE DRVCK1     NO
0267                    *
0268   0121 4F                CLR A          DRIVE OK
0269   0122 0C                CLC
0270   0123 39                RTS
0271                    *
0272   0124 0D          DRVCK1 SEC
0273   0125 86 0A             LDA A #10      RETURN ERROR=10
0274   0127 39                RTS
0275                    *
0276                           END
```

```
.RDSEC 002C RN
.WTSEC 0097 RN
@INTDK 0000 RN
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
CHAIN  243A M
CLEAR  0080
CLOSE  2369 M
CLRERF 000A
CMDCTL EC03
CMDDAT EC02
CMPC   231B M
CMWC   2572 M
DELETE 2420 M
DIV16  2524 M
DRIVCK 011A R
DRVCK1 0124 R
ERFRST 0104 R
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBSCT 000B
FCBSTA 0005
FCBTRK 000A
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
GETBF0 0044 R
GETBF1 004A R
GETBF2 0060 R
GETBF3 0068 R
GETBF4 006A R
GETBUF 003A R
GETDR  23EC M
GTCMD  24F0 M
ICOMDR 0000 RN
INCTL  EC01
INDAT  EC00
INDEX  24BC M
INITDK 253E M
IOHDR  2335 M
LDTRAD 0010
LDUS   0020
LDWBF  0030
LOADB  246E M
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
OUTCM1 00FB R
OUTCMD 00F3 R
OUTCTL EC07
OUTDAT EC06
PRTERR 2454 M
PRTMSG 250A M
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
QUIT   008D R
RCBDEF 258C M
RDCRC  0006
READ   23B8 M
READX  0002
REWIND 2384 M
SEEK   0008
SEEKTO 000C
SEEKTK 010C R
SHFTRB 0040
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
TABX   219C M
TXAB   2183 M
UA     0006
UXH    0007
WRITE  23D2 M
WRITEX 0004
WRTBF0 00A9 R
WRTBF1 00C3 R
WRTBF2 00C9 R
WRTBF3 00E4 R
WRTBUF 00A5 R
XABX   21B5 M
XMITUS 00E7 R
```

```
0001    0000 0000     N       NAM ASSIGN
0002                  *
0003                  * DEVICE ASSIGNMENT TRANSIENT ROUTINE
0004                  * SYNTAX: ASSIGN DEV1=DEV2
0005                  * TERMINATE WITH AN ESCAPE,CR
0006                  *
0007                          BASEQU
0008 +  0000 0020     DESCRA EQU $20  DESCRIPTOR ADDRESS(2)
0009 +  0000 0022     DESCRC EQU $22  DESCRIPTOR COUNT
0010 +  0000 0023     CUCHAR EQU $23  CURRENT CHAR (2)
0011 +  0000 0025     RC     EQU $25  TOKEN RETURN CODE
0012 +  0000 0026     CLASS  EQU $26  TOKEN CLASS
0013 +  0000 0027     VALUE  EQU $27  BIN VALUE/TRANSFER ADDRESS (2)
0014 +  0000 0029     FCBCHN EQU $29  TOP OF FCB CHAIN (2)
0015 +  0000 002B     FRETAB EQU $2B  DISK FREE SPACE POINTER (8)
0016 +  0000 0033     BMEM   EQU $33  START OF TRANSIENT AREA(2)
0017 +  0000 0035     EMEM   EQU $35  END OF TRANSIENT AREA (2)
0018 +  0000 0037     CMEM   EQU $37  NEXT AVAIL TRANSIENT AREA (2)
0019 +  0000 0039     BS     EQU $39  BACKSPACE CHAR
0020 +  0000 003A     DL     EQU $3A  DELETE LINE CHAR
0021 +  0000 003B     DP     EQU $3B  DEPTH; LINES/PAGE
0022 +  0000 003C     DPCNT  EQU $3C  DEPTH TEMP
0023 +  0000 003D     WD     EQU $3D  WIDTH; CHARS/LINE
0024 +  0000 003E     NL     EQU $3E  NULL COUNT
0025 +  0000 003F     TB     EQU $3F  TAB CHAR
0026 +  0000 0040     DX     EQU $40  DUPLEX; FF=H, 00=F
0027 +  0000 0041     EJ     EQU $41  EJECT COUNT
0028 +  0000 0042     PS     EQU $42  PAUSE; 00=YES
0029 +  0000 0043     ES     EQU $43  ESCAPE CHAR
0030 +  0000 0044     LDP    EQU $44  DEPTH LINES/PAGE
0031 +  0000 0045     LDPCNT EQU $45  DEPTH TEMP
0032 +  0000 0046     LWD    EQU $46  WIDTH CHARS/LINE
0033                  *
0034    0000 B7 011B R        STA A PDTAB          SAVE POINTER TO PDTAB
0035    0003 F7 011C R        STA B PDTAB+1
0036                  *
0037                  ASSNO   NXTOK          GET DEVICE NAME
0038 +   0006 3F               SWI
0039 +   0007 2F               FCB 47
0040    0008 DE 20            LDX DESCRA
0041    000A A6 00            LDA A 0,X
0042    000C 91 43            CMP A ES       ESCAPE?
0043    000E 26 09            BNE ASSN2      NO
0044                  *
0045    0010 39               RTS            YES, DONE
0046                  *
0047    0011 CE 00F1 R ASSN1  LDX #MSGA      "SYNTAX ERROR"
0048                          PRTMSG
0049 +   0014 3F               SWI
0050 +   0015 31               FCB 49
0051    0016 7E 007E R        JMP ASSNXT
0052                  *
0053    0019 D6 25    ASSN2   LDA B RC
0054    001B C1 01            CMP B #1       NAME?
0055    001D 26 F2            BNE ASSN1      NO
0056                  *
0057    001F D6 22            LDA B DESCRC   GET LENGTH
0058    0021 C1 03            CMP B #3       VALID?
0059    0023 26 EC            BNE ASSN1      NO
0060                  *
0061                  * SAVE DEV1 NAME
0062                  *
0063    0025 CE 011D R        LDX #DEV1      "TOO"
0064                          PSHX
0065 +   0028 3F               SWI
0066 +   0029 05               FCB 5
0067    002A DE 20            LDX DESCRA     "FROM"
0068                          PSHX
0069 +   002C 3F               SWI
0070 +   002D 05               FCB 5
0071    002E C6 03            LDA B #3       COUNT
0072                          MOVC
0073 +   0030 3F               SWI
0074 +   0031 11               FCB 17
0075    0032 31               INS            FIX STACK
0076    0033 31               INS
0077    0034 31               INS
0078    0035 31               INS
0079                          NXTOK          GET "="
0080 +   0036 3F               SWI
0081 +   0037 2F               FCB 47
0082    0038 D6 25            LDA B RC
0083    003A C1 3D            CMP B #'=      OK?
0084    003C 26 D3            BNE ASSN1      NO
0085                  *
0086                          NXTOK          GET DEV2 NAME
0087 +   003E 3F               SWI
0088 +   003F 2F               FCB 47
0089    0040 D6 25            LDA B RC
0090    0042 C1 01            CMP B #1       NAME?
0091    0044 26 CB            BNE ASSN1      NO
0092                  *
0093    0046 D6 22            LDA B DESCRC
0094    0048 C1 03            CMP B #3       VALID?
0095    004A 26 C5            BNE ASSN1      NO
0096                  *
0097                  * SAVE DEV2 NAME
0098                  *
0099    004C CE 0120 R        LDX #DEV2      "TOO"
0100                          PSHX
0101 +   004F 3F               SWI
0102 +   0050 05               FCB 5
0103    0051 DE 20            LDX DESCRA     "FROM"
0104                          PSHX
0105 +   0053 3F               SWI
0106 +   0054 05               FCB 5
0107    0055 C6 03            LDA B #3       COUNT
0108                          MOVC
0109 +   0057 3F               SWI
0110 +   0058 11               FCB 17
0111    0059 31               INS
0112    005A 31               INS
0113    005B 31               INS
0114    005C 31               INS
0115                  *
0116                  * SEE IF DEV1=DEV2
0117                  *
0118    005D CE 011D R        LDX #DEV1
0119                          PSHX
0120 +   0060 3F               SWI
0121 +   0061 05               FCB 5
```

```
0122   0062 CE 0120 R          LDX #DEV2
0123                           PSHX
0124 +  0065 3F                 SWI
0125 +  0066 05                 FCB 5
0126   0067 C6 03              LDA B #3
0127                           CMPC
0128 +  0069 3F                 SWI
0129 +  006A 12                 FCB 18
0130   006B 31                 INS
0131   006C 31                 INS
0132   006D 31                 INS
0133   006E 31                 INS
0134   006F 26 25              BNE ASSN4      NO
0135                   *
0136                   * SEARCH PDTAB FOR DEV1
0137                   *
0138   0071 CE 011D R          LDX #DEV1
0139   0074 BD 00BB R          JSR PDSRCH
0140   0077 24 13              BCC ASSN3      OK
0141                   *
0142   0079 CE 00FE R PMSGB    LDX #MSGB      INVALID NAME
0143                           PRTMSG
0144 +  007C 3F                 SWI
0145 +  007D 31                 FCB 49
0146                   *
0147   007E CE 0112 R ASSNXT   LDX #MSGC
0148                           PRTMSG
0149 +  0081 3F                 SWI
0150 +  0082 31                 FCB 49
0151                           GTCMD
0152 +  0083 3F                 SWI
0153 +  0084 30                 FCB 48
0154   0085 DE 20              LDX DESCRA
0155   0087 DF 23              STX CUCHAR
0156   0089 7E 0006 R          JMP ASSNO
0157                   *
0158                   * DEV1=DEV2  X=A(ADDR1) OF DEV1
0159                   *
0160   008C A6 02      ASSN3   LDA A 2,X
0161   008E A7 00              STA A 0,X
0162   0090 A6 03              LDA A 3,X
0163   0092 A7 01              STA A 1,X
0164   0094 20 E8              BRA ASSNXT
0165                   *
0166                   * DEV1 .NE. DEV2
0167                   * SEARCH PDTAB FOR DEV2
0168                   *
0169   0096 CE 0120 R ASSN4   LDX #DEV2
0170   0099 BD 00BB R          JSR PDSRCH
0171   009C 24 02              BCC ASSN5      FOUND
0172                   *
0173   009E 20 D9              BRA PMSGB      ERROR
0174                   *
0175   00A0 EE 02      ASSN5   LDX 2,X        GET DEV2 ADDR2
0176                           PSHX           SAVE
0177 +  00A2 3F                 SWI
0178 +  00A3 05                 FCB 5
0179                   *
0180   00A4 CE 011D R          LDX #DEV1
0181   00A7 BD 00BB R          JSR PDSRCH
0182   00AA 24 02              BCC ASSN6      FOUND
0183                   *
0184   00AC 20 CB              BRA PMSGB      ERROR
0185                   *
0186                   * X= A(ADDR1) DEV1
0187                   *
0188                   ASSN6   TXAB           SAVE IN A,B
0189 +  00AE 3F                 SWI
0190 +  00AF 02                 FCB 2
0191                           PULX           GET DEV2, ADDR2
0192 +  00B0 3F                 SWI
0193 +  00B1 06                 FCB 6
0194                           XABX           SWITCH
0195 +  00B2 3F                 SWI
0196 +  00B3 04                 FCB 4
0197   00B4 A7 00              STA A 0,X      DO ASSIGN
0198   00B6 E7 01              STA B 1,X
0199   00B8 7E 007E R          JMP ASSNXT
0200                   *
0201                   * SEARCH PDTAB
0202                   *
0203                   PDSRCH  PSHX           STACK DEV ADDRESS
0204 +  00BB 3F                 SWI
0205 +  00BC 05                 FCB 5
0206   00BD FE 011B R          LDX PDTAB      POINT TO PDTAB
0207                   *
0208                   PDSRCA  PSHX           STACK PDTAB PTR
0209 +  00C0 3F                 SWI
0210 +  00C1 05                 FCB 5
0211   00C2 C6 03              LDA B #3       COUNT
0212                           CMPC
0213 +  00C4 3F                 SWI
0214 +  00C5 12                 FCB 18
0215   00C6 27 23              BEQ PDSRCB     FOUND
0216                   *
0217                   * NO MATCH
0218                   *
0219                           PULX           GET PDTAB PTR
0220 +  00C8 3F                 SWI
0221 +  00C9 06                 FCB 6
0222                           ADDBX          FIX POINTER
0223 +  00CA 3F                 SWI
0224 +  00CB 0A                 FCB 10
0225   00CC 08                 INX
0226   00CD 08                 INX
0227   00CE 08                 INX
0228   00CF 08                 INX
0229                           PSHX           STACK PDTAB POINTER
0230 +  00D0 3F                 SWI
0231 +  00D1 05                 FCB 5
0232   00D2 30                 TSX
0233   00D3 EE 02              LDX 2,X        GET DEVICE ADDR POINTER
0234   00D5 86 03              LDA A #3
0235   00D7 10                 SBA
0236                           SUBAX          RESET
0237 +  00D8 3F                 SWI
0238 +  00D9 0D                 FCB 13
0239                           TXAB           SAVE
0240 +  00DA 3F                 SWI
0241 +  00DB 02                 FCB 2
0242   00DC 30                 TSX
0243   00DD A7 02              STA A 2,X
```

```
0244   00DF E7 03          STA B 3,X
0245                       PULX            GET PDTAB POINTER
0246 +  00E1 3F             SWI
0247 +  00E2 06             FCB 6
0248   00E3 6D 00          TST 0,X         END OF TABLE?
0249   00E5 26 D9          BNE PDSRCA      NO
0250                *
0251                * YES NOT IN TABLE
0252                *
0253   00E7 31             INS             FIX STACK
0254   00E8 31             INS
0255   00E9 0D             SEC             SET RC
0256   00EA 39             RTS
0257                *
0258                PDSRCB PULX            GET PDTAB POINTER
0259 +  00EB 3F             SWI
0260 +  00EC 06             FCB 6
0261   00ED 31             INS             FIX STACK
0262   00EE 31             INS
0263   00EF 0C             CLC             SET RC
0264   00F0 39             RTS
0265                *
0266                *
0267   00F1 53     MSGA    FCC 'SYNTAX ERROR'
0268   00FD 0D             FCB $0D
0269                *
0270   00FE 49     MSGB    FCC 'INVALID DEVICE NAME'
0271   0111 0D             FCB $0D
0272                *
0273   0112 41     MSGC    FCC 'ASSIGN- '
0274   011A 04             FCB 4
0275                *
0276   011B 0002   PDTAB   RMB 2
0277   011D 0003   DEV1    RMB 3
0278   0120 0003   DEV2    RMB 3
0279                *
0280                        END
```

```
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
ASSIGN 0000 RN
ASSN0  0006 R
ASSN1  0011 R
ASSN2  0019 R
ASSN3  008C R
ASSN4  0096 R
ASSN5  00A0 R
ASSN6  00AE R
ASSNXT 007E R
BASEQU 2A2A M
BMEM   0033
BS     0039
CHAIN  243A M
CLASS  0026
CLOSE  2369 M
CMEM   0037
CMPC   231B M
CMWC   2572 M
CUCHAR 0023
DELETE 2420 M
DESCRA 0020
DESCRC 0022
DEV1   011D R
DEV2   0120 R
DIV16  2524 M
DL     003A
DP     003B
DPCNT  003C
DX     0040
EJ     0041
EMEM   0035
ES     0043
FCBCHN 0029
FCBDEF 2650 M
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
FRETAB 002B
GETDR  23EC M
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
IOHDR  2335 M
LDP    0044
LDPCNT 0045
LOADB  246E M
LWD    0046
MOVC   2301 M
MOVS   24A2 M
MSGA   00F1 R
MSGB   00FE R
MSGC   0112 R
MUL16  22E7 M
MUL8   22CD M
NL     003E
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
PDSRCA 00C0 R
PDSRCB 00EB R
PDSRCH 00BB R
PDTAB  011B R
PMSGB  0079 R
PRTERR 2454 M
PRTMSG 250A M
PS     0042
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
RC     0025
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
TABX   219C M
TB     003F
TXAB   2183 M
VALUE  0027
WD     003D
WRITE  23D2 M
XABX   21B5 M
```

```
0001   0000 0000     N        NAM BOOT
0002                  *
0003                  * ICOM CP/68 BOOTSTRAP PROGRAM
0004                  * ASSUMES SYSTEM FILE LINKED AS FOLLOWS:
0005                  *
0006                  *      TRACK 0, SECTOR 3, BYTE 122-FIRST TRACK
0007                  *                             123-FIRST SECTOR
0008                  *                             124-LAST TRACK
0009                  *                             125-LAST SECTOR
0010                  *                             126,7 FREE-SPACE HEADER
0011                  *
0012                  * BOOTS SYSTEM FROM DRIVE 0:
0013                  *
0014                  * DEFINE DISK-DRIVE INTERFACE ADDRESSING
0015                  *
0016   0000 EC00      INDAT   EQU $EC00
0017   0000 EC01      INCTL   EQU $EC01
0018   0000 EC02      CMDDAT EQU $EC02
0019   0000 EC03      CMDCTL EQU $EC03
0020   0000 EC06      OUTDAT EQU $EC06
0021   0000 EC07      OUTCTL EQU $EC07
0022                  *
0023                  * NOTE: ALL VARIABLES IN COMMON, CODE IS ROM-ABLE
0024                  *
0025   0000 0000      C        CMN STACK,16
0026   0000 0010      C        CMN BUFFER,128
0027   0000 0090      C        CMN FTS,2
0028   0000 0092      C        CMN LTS,2
0029   0000 0094      C        CMN PTS,2
0030   0000 0096      C        CMN INDEX,2
0031   0000 0098      C        CMN SAVEX,2
0032   0000 009A      C        CMN ADDRES,2
0033   0000 009C      C        CMN FCNT,1
0034                  *
0035                  * ERROR JUMP VECTOR
0036                  *
0037   0000 E113      ERROR   EQU $E113
0038                  *
0039                  * BEGIN BOOT HERE
0040                  *
0041   0000 8E 000F C START   LDS #STACK+15 INIT.  STACK POINTER
0042   0003 7F EC01           CLR INCTL     INIT.  INTERFACE
0043   0006 7F EC03           CLR CMDCTL
0044   0009 7F EC07           CLR OUTCTL
0045   000C 7F EC00           CLR INDAT
0046   000F 86 FF             LDA A #$FF
0047   0011 B7 EC02           STA A CMDDAT
0048   0014 B7 EC06           STA A OUTDAT
0049   0017 86 04             LDA A #$04
0050   0019 B7 EC01           STA A INCTL
0051   001C B7 EC07           STA A OUTCTL
0052   001F 86 2C             LDA A #$2C
0053   0021 B7 EC03           STA A CMDCTL
0054   0024 86 80             LDA A #$80
0055   0026 B7 EC02           STA A CMDDAT  ISSUE 'CLEAR' COMMAND
0056   0029 86 0C             LDA A #$0C    ISSUE 'SEEK TRACK 0' COMMAND
0057   002B BD 0147 R         JSR OUTCMD
0058                  *
0059                  * NOW GET SYSTEM LINK INFORMATION
0060                  *
0061   002E 86 03             LDA A #3
0062   0030 C6 00             LDA B #0      TRACK 0, SECTOR 3
0063   0032 CE 0010 C         LDX #BUFFER
0064   0035 BD 00DD R         JSR RDSEC     READ LINK SECTOR
0065   0038 CE 0010 C         LDX #BUFFER
0066   003B A6 7A             LDA A 122,X   GET FIRST T/S
0067   003D E6 7B             LDA B 123,X
0068   003F B7 0090 C         STA A FTS
0069   0042 F7 0091 C         STA B FTS+1
0070   0045 A6 7C             LDA A 124,X   GET LAST T/S
0071   0047 E6 7D             LDA B 125,X
0072   0049 B7 0092 C         STA A LTS
0073   004C F7 0093 C         STA B LTS+1
0074   004F CE 0014 C         LDX #BUFFER+4
0075   0052 FF 0096 C         STX INDEX     INIT.  BUFFER INDEX
0076   0055 B6 0091 C         LDA A FTS+1
0077   0058 F6 0090 C         LDA B FTS
0078   005B B7 0095 C         STA A PTS+1   INIT.  PRESENT T/S
0079   005E F7 0094 C         STA B PTS
0080   0061 CE 0010 C         LDX #BUFFER
0081   0064 BD 00DD R         JSR RDSEC     READ FIRST SECTOR
0082                  *
0083                  * NOW LOAD SYSTEM FILE INTO MEMORY
0084                  *
0085   0067 8D 3A     BOOT1   BSR GETBYT    GET A DATA BYTE FROM FILE
0086   0069 81 16             CMP A #$16    TRANSFER-ADDRESS?
0087   006B 26 0C             BNE BOOT2     NO
0088                  *
0089   006D 8D 34             BSR GETBYT
0090   006F B7 009A C         STA A ADDRES  GET TRANSFER ADDRESS
0091   0072 8D 2F             BSR GETBYT
0092   0074 B7 009B C         STA A ADDRES+1
0093   0077 20 EE             BRA BOOT1     GET NEW DATA FRAME
0094                  *
0095   0079 81 02     BOOT2   CMP A #$02    DATA FRAME?
0096   007B 26 21             BNE BOOT4     NO
0097                  *
0098   007D 8D 24             BSR GETBYT
0099   007F B7 0098 C         STA A SAVEX   GET ADDRESS
0100   0082 8D 1F             BSR GETBYT
0101   0084 B7 0099 C         STA A SAVEX+1
0102   0087 8D 1A             BSR GETBYT
0103   0089 B7 009C C         STA A FCNT    GET FRAME COUNTER
0104                  *
0105   008C 8D 15     BOOT3   BSR GETBYT    GET DATA BYTE
0106   008E FE 0098 C         LDX SAVEX
0107   0091 A7 00             STA A 0,X     STORE BYTE
0108   0093 08                INX
0109   0094 FF 0098 C         STX SAVEX
0110   0097 7A 009C C         DEC FCNT      COUNT DOWN
0111   009A 26 F0             BNE BOOT3
0112                  *
0113   009C 20 C9             BRA BOOT1     GET NEW DATA FRAME
0114                  *
0115   009E FE 009A C BOOT4   LDX ADDRES    GET TRANSFER ADDRESS
0116   00A1 6E 00             JMP 0,X       GO THERE
0117                  *
0119                  * READ A DATA BYTE FROM SYSTEM FILE
0120                  * RETURN BYTE IN 'A' REGISTER
0121                  *
0122   00A3 FE 0096 C GETBYT LDX INDEX
```

```
0123  00A6 8C 0090 C        CPX #BUFFER+128     NEED NEW SECTOR?
0124  00A9 27 07            BEQ GETSEC          YES
0125                  *
0126  00AB A6 00            LDA A 0,X           GET BYTE
0127  00AD 08               INX
0128  00AE FF 0096 C        STX INDEX           MOVE POINTER
0129  00B1 39               RTS
0130                  *
0131  00B2 F6 0094 C GETSEC LDA B PTS
0132  00B5 B6 0095 C        LDA A PTS+1         CHECK FOR LAST SECTOR
0133  00B8 B1 0093 C        CMP A LTS+1
0134  00BB 26 07            BNE GETS2           NOT LAST
0135                  *
0136  00BD F1 0092 C        CMP B LTS
0137  00C0 26 02            BNE GETS2           NOT LAST
0138                  *
0139  00C2 20 DA            BRA BOOT4           EOF-GO TO TRANSFER ADDRESS
0140                  *
0141  00C4 CE 0010 C GETS2  LDX #BUFFER
0142  00C7 E6 00            LDA B 0,X           GET FORWARD T/S LINK
0143  00C9 A6 01            LDA A 1,X
0144  00CB F7 0094 C        STA B PTS           UPDATE PRESENT T/S
0145  00CE B7 0095 C        STA A PTS+1
0146  00D1 8D 0A            BSR RDSEC           READ NEW SECTOR
0147  00D3 CE 0014 C        LDX #BUFFER+4
0148  00D6 A6 00            LDA A 0,X           GET DATA BYTE
0149  00D8 08               INX
0150  00D9 FF 0096 C        STX INDEX           RE-INIT. INDEX
0151  00DC 39               RTS
0152                  *
0154                  * SINGLE-SECTOR READ ROUTINE
0155                  *
0156                  *     DRIVE=0
0157                  *     TRACK='B'
0158                  *     SECTOR='A'
0159                  *     BUFFER='X'
0160                  *
0161  00DD 36         RDSEC PSH A               SAVE SECTOR
0162  00DE 86 0A            LDA A #$0A          RESET ERROR FLAGS
0163  00E0 B7 EC02          STA A CMDDAT
0164  00E3 32               PUL A
0165  00E4 B7 EC06          STA A OUTDAT        OUTPUT SECTOR
0166  00E7 86 20            LDA A #$20          LOAD U/S COMMAND
0167  00E9 B7 EC02          STA A CMDDAT
0168  00EC B6 EC00          LDA A INDAT         CHECK DRIVE STATUS
0169  00EF 84 20            AND A #$20          DISK READY?
0170  00F1 27 03            BEQ *+5             YES
0171                  *
0172  00F3 7E E113          JMP ERROR           NO
0173                  *
0174  00F6 17               TBA
0175  00F7 B7 EC06          STA A OUTDAT        OUTPUT TRACK
0176  00FA 86 10            LDA A #$10          ISSUE 'LOAD TRACK ADDRESS' COMMAND
0177  00FC B7 EC02          STA A CMDDAT
0178  00FF 86 08            LDA A #$08          ISSUE 'SEEK' COMMAND
0179  0101 8D 44            BSR OUTCMD
0180  0103 C6 05            LDA B #5            FIVE RETRIES
0181                  *
0182  0105 86 02      RDSEC1 LDA A #$02
0183  0107 8D 3E            BSR OUTCMD          ISSUE 'READ' COMMAND
0184  0109 B6 EC00          LDA A INDAT         CHECK STATUS OF DRIVE
0185  010C 85 08            BIT A #$08          CRC ERROR?
0186  010E 27 0B            BEQ RDSEC2          NO
0187                  *
0188  0110 86 0A            LDA A #$0A          RESET ERROR FLAGS
0189  0112 B7 EC02          STA A CMDDAT
0190  0115 5A               DEC B
0191  0116 26 ED            BNE RDSEC1          RETRY READ
0192                  *
0193  0118 7E E113          JMP ERROR           FATAL ERROR
0194                  *
0195  011B 85 80      RDSEC2 BIT A #$80         DDAM?
0196  011D 27 03            BEQ RDSEC3          NO
0197                  *
0198  011F 7E E113          JMP ERROR           YES, FATAL ERROR
0199                  *
0200  0122 C6 80      RDSEC3 LDA B #128         128 BYTES IN SECTOR
0201                  *
0202  0124 86 3C      RDSEC4 LDA A #$3C
0203  0126 B7 EC03          STA A CMDCTL        INIT. CMD. CNTL. REG.
0204  0129 86 40            LDA A #$40
0205  012B B7 EC02          STA A CMDDAT        ISSUE 'READ-DATA' COMMAND
0206  012E B6 EC00          LDA A INDAT
0207  0131 36               PSH A               SAVE DATA BYTE
0208  0132 86 2C            LDA A #$2C
0209  0134 B7 EC03          STA A CMDCTL        RESET CMD. CNTL. REG.
0210  0137 86 40            LDA A #$40
0211  0139 B7 EC02          STA A CMDDAT        STROBE READ BUFFER
0212  013C 7F EC02          CLR CMDDAT
0213  013F 32               PUL A               RECOVER DATA BYTE
0214  0140 A7 00            STA A 0,X
0215  0142 08               INX
0216  0143 5A               DEC B               COUNT DOWN
0217  0144 26 DE            BNE RDSEC4
0218                  *
0219  0146 39               RTS                 DONE!
0220                  *
0221                  *
0222                  * OUTPUT COMMAND FROM 'A' REGISTER
0223                  *
0224  0147 36         OUTCMD PSH A              SAVE COMMAND
0225  0148 B6 EC00          LDA A INDAT         CLEAR 'BUSY'
0226  014B 32               PUL A
0227  014C B7 EC02          STA A CMDDAT        ISSUE COMMAND
0228  014F B6 EC01  OUTCD1  LDA A INCTL         DONE?
0229  0152 2A FB            BPL OUTCD1          WAIT FOR DONE
0230                  *
0231  0154 B6 EC00          LDA A INDAT         CLEAR 'BUSY'
0232  0157 39               RTS
0233                  *
0234                        END
```

```
ADDRES 009A C
BOOT   0000 RN
BOOT1  0067 R
BOOT2  0079 R
BOOT3  008C R
BOOT4  009E R
BUFFER 0010 C
CMDCTL EC03
CMDDAT EC02
ERROR  E113
FCNT   009C C
FTS    0090 C
GETBYT 00A3 R
GETS2  00C4 R
GETSEC 00B2 R
INCTL  EC01
INDAT  EC00
INDEX  0096 C
LTS    0092 C
OUTCD1 014F R
OUTCMD 0147 R
OUTCTL EC07
OUTDAT EC06
PTS    0094 C
RDSEC  00DD R
RDSEC1 0105 R
RDSEC2 011B R
RDSEC3 0122 R
RDSEC4 0124 R
SAVEX  0098 C
STACK  0000 C
START  0000 R
```

```
0001   0000 0000    N       NAM DELFILE
0002                        *
0003                        * TRANSIENT 'DELETE' COMMAND PROCESSOR
0004                        * FOR CP/68 OPERATING SYSTEM
0005                        *
0006                        * BLOCK ADDRESSING DEFINITIONS
0007                        *
0008                              FCBDEF
0009 +  0000 0000           FCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0010 +  0000 0002           FCBGDT EQU 2          GENERIC DEVICE TYPE
0011 +  0000 0005           FCBSTA EQU 5          STATUS
0012 +  0000 0006           FCBDTT EQU 6          DATA TRANSFER TYPE
0013 +  0000 0007           FCBDBA EQU 7          DATA BUFFER ADDRESS
0014 +  0000 0009           FCBDRV EQU 9          DRIVE NUMBER
0015 +  0000 000A           FCBTRK EQU 10         TRACK NUMBER
0016 +  0000 000B           FCBSCT EQU 11         SECTOR NUMBER
0017 +  0000 000C           FCBFWD EQU 12         FWD LINK TRACK/SECTOR
0018 +  0000 000E           FCBBAK EQU 14         BACK LINK TRACK/SECTOR
0019 +  0000 0010           FCBNAM EQU 16         FILE NAME (8.3+EOT=13))
0020 +  0000 001D           FCBTYP EQU 29         FILE TYPE
0021 +  0000 001E           FCBACS EQU 30         FILE ACCESS CODE
0022 +  0000 001F           FCBFTS EQU 31         FIRST TRACK/SECTOR
0023 +  0000 0021           FCBLTS EQU 33         LAST TRACK/SECTOR
0024 +  0000 0023           FCBNMS EQU 35         NUMBER OF SECTORS
0025 +  0000 0025           FCBNFB EQU 37         NEXT FCB IN ACTIVE CHAIN
0026 +  0000 0027           FCBIND EQU 39         INDEX INTO DATA BUFFER
0027 +  0000 0029           FCBSCF EQU 41         SPACE COMPRESSION FLAG
0028                              FIBDEF
0029 +  0000 0000           FIBNAM EQU 0          FILE NAME (8.3 + EOT=13)
0030 +  0000 000D           FIBTYP EQU 13         FILE TYPE
0031 +  0000 000E           FIBACS EQU 14         FILE ACCESS CODE
0032 +  0000 000F           FIBFTS EQU 15         FIRST TRACK/SECTOR
0033 +  0000 0011           FIBLTS EQU 17         LAST TRACK/SECTOR
0034 +  0000 0013           FIBNMS EQU 19         NUMBER OF SECTORS
0035                        *
0036                        * BASE-PAGE EQUATES
0037                        *
0038   0000 0020            DESCRA EQU $20
0039   0000 0022            DESCRC EQU $22
0040   0000 0023            CUCHAR EQU $23
0041   0000 0025            RC     EQU $25
0042   0000 0026            CLASS  EQU $26
0043   0000 0027            VALUE  EQU $27
0044   0000 0043            ESCAPE EQU $43
0045                        *
0046                        * DISK ATTRIBUTES
0047                        *
0048   0000 0080            SECSIZ EQU 128        128 BYTES/SECTOR
0049                        *
0050                        * FCB FOR TRANSIENT
0051                        *
0052   0000 0002            SYSFCB RMB 2
0053   0002 44                     FCC 'DSK'
0054   0005 0002                   RMB 2
0055   0007 002A    R              FDB BUFFER
0056   0009 0021                   RMB 33
0057   002A 0080            BUFFER RMB SECSIZ
0058                        *
0059                        * TEMPORARY STORAGE FOR FILE NAME
0060                        *
```

```
0061   00AA 0002       SAVEX  RMB 2
0062   00AC 000C       TEMP   RMB 12
0063                   *
0064   00B8 CE 0000 R  DEL0   LDX #SYSFCB
0065   00BB 6F 09             CLR FCBDRV,X          DEFAULT DRIVE=0
0066   00BD 6F 06             CLR FCBDTT,X          INPUT
0067                          NXTOK                 GET TOKEN FROM CLI
0068 +  00BF 3F                SWI
0069 +  00C0 2F                FCB 47
0070   00C1 DE 20             LDX DESCRA
0071   00C3 A6 00             LDA A 0,X             CHECK FOR 'ESCAPE'
0072   00C5 91 43             CMP A ESCAPE
0073   00C7 26 01             BNE DEL1              NOT ESCAPE
0074                   *
0075   00C9 39                RTS                   'ESCAPE'...DONE
0076                   *
0077   00CA D6 25      DEL1   LDA B RC              CHECK RC OF TOKEN
0078   00CC C1 03             CMP B #3              NUMBER?
0079   00CE 26 34             BNE DEL2              NO
0080                   *
0081   00D0 7D 0027           TST VALUE             VALID DRIVE NO.?
0082   00D3 26 0D             BNE DEL1A             NO
0083                   *
0084   00D5 96 28             LDA A VALUE+1
0085   00D7 81 03             CMP A #3              VALID DRIVE NO.?
0086   00D9 22 07             BHI DEL1A             NO
0087                   *
0088   00DB CE 0000 R         LDX #SYSFCB
0089   00DE A7 09             STA A FCBDRV,X        SET DRIVE NO. IN FCB
0090   00E0 20 16             BRA DEL1B
0091                   *
0092   00E2 CE 00EA R  DEL1A  LDX #NUMBER           NUMBER ERROR
0093                          PRTMSG
0094 +  00E5 3F                SWI
0095 +  00E6 31                FCB 49
0096   00E7 7E 0221 R         JMP DELNXT            GET NEW CLI
0097                   *
0098   00EA 20         NUMBER FCC ' NUMBER ERROR'
0099   00F7 0D                FCB $0D
0100                   *
0101                   DEL1B  NXTOK                 GET A TOKEN
0102 +  00F8 3F                SWI
0103 +  00F9 2F                FCB 47
0104   00FA D6 25             LDA B RC              CHECK RC
0105   00FC C1 3A             CMP B #':             COLON?
0106   00FE 26 E2             BNE DEL1A             IF NOT, ERROR
0107                   *
0108                          NXTOK                 GET A TOKEN
0109 +  0100 3F                SWI
0110 +  0101 2F                FCB 47
0111   0102 D6 25             LDA B RC              CHECK RC
0112   0104 C1 01      DEL2   CMP B #1              UNAMBIG. NAME?
0113   0106 27 1A             BEQ DEL3              YES
0114                   *
0115   0108 C1 02             CMP B #2              AMBIG. NAME?
0116   010A 27 16             BEQ DEL3              YES
0117                   *
0118   010C CE 0114 R  DEL2A  LDX #FORMAT           FORMAT ERROR
0119                          PRTMSG
0120 +  010F 3F                SWI
0121 +  0110 31                FCB 49
0122   0111 7E 0221 R         JMP DELNXT            GET NEW CLI
0123                   *
0124   0114 20         FORMAT FCC ' FORMAT ERROR'
0125   0121 0D                FCB $0D
0126                   *
0127   0122 DE 20      DEL3   LDX DESCRA
0128   0124 FF 002A R         STX BUFFER            SAVE POINTER TO NAME
0129   0127 96 22             LDA A DESCRC
0130   0129 B7 002C R         STA A BUFFER+2        SAVE LENGTH OF NAME
0131                          NXTOK                 GET A TOKEN
0132 +  012C 3F                SWI
0133 +  012D 2F                FCB 47
0134   012E D6 25             LDA B RC              CHECK RC
0135   0130 C1 2E             CMP B #'.             PERIOD?
0136   0132 26 D8             BNE DEL2A             IF NOT, ERROR
0137                   *
0138   0134 7C 002C R         INC BUFFER+2          COUNT PERIOD
0139                          NXTOK                 GET A TOKEN
0140 +  0137 3F                SWI
0141 +  0138 2F                FCB 47
0142   0139 D6 25             LDA B RC              CHECK RC
0143   013B C1 01             CMP B #1              UNAMBIG. NAME?
0144   013D 27 04             BEQ DEL4              YES
0145                   *
0146   013F C1 02             CMP B #2              AMBIG. NAME?
0147   0141 26 C9             BNE DEL2A             IF NOT, ERROR
0148                   *
0149   0143 D6 22      DEL4   LDA B DESCRC          GET LENGTH OF EXT
0150   0145 FB 002C R         ADD B BUFFER+2        TOTAL LENGTH
0151   0148 CE 00AC R         LDX #TEMP
0152                          PSHX                  POINT TO BUFFER
0153 +  014B 3F                SWI
0154 +  014C 05                FCB 5
0155   014D FE 002A R         LDX BUFFER            POINT TO FILE-NAME
0156                          PSHX
0157 +  0150 3F                SWI
0158 +  0151 05                FCB 5
0159                          FMTS                  FORMAT NAME INTO TEMP BUFFER
0160 +  0152 3F                SWI
0161 +  0153 34                FCB 52
0162   0154 31                INS
0163   0155 31                INS                   CLEAN STACK
0164   0156 31                INS
0165   0157 31                INS
0166   0158 C1 02             CMP B #2              ERROR?
0167   015A 27 B0             BEQ DEL2A             YES
0168                   *
0169   015C CE 0000 R         LDX #SYSFCB
0170   015F 6F 29             CLR FCBSCF,X          CLEAR 'FILE-FOUND' MARK
0171                          OPEND                 OPEN DIRECTORY
0172 +  0161 3F                SWI
0173 +  0162 17                FCB 23
0174   0163 A6 05      DEL4A  LDA A FCBSTA,X        CHECK STATUS
0175   0165 27 28             BEQ DEL5              GOOD?
0176                   *
0177   0167 81 01             CMP A #1              END OF DIRECTORY?
0178   0169 26 1F             BNE DEL4B             NO
0179                   *
0180   016B 6D 29             TST FCBSCF,X          FILE FOUND DURING SEARCH?
0181   016D 27 03             BEQ *+5               NO, ERROR
0182                   *
```

```
0183   016F 7E 0221 R        JMP DELNXT            YES, GET NEW CLI LINE
0184                 *
0185   0172 CE 017A R        LDX #FNFND            FILE-NOT-FOUND ERROR
0186                         PRTMSG
0187 +  0175 3F               SWI
0188 +  0176 31               FCB 49
0189   0177 7E 0221 R        JMP DELNXT
0190                 *
0191   017A 20        FNFND  FCC ' FILE NOT FOUND'
0192   0189 0D               FCB $0D
0193                 *
0194                 DEL4B   PRTERR               PRINT ERROR MESSAGE
0195 +  018A 3F               SWI
0196 +  018B 1E               FCB 30
0197   018C 7E 0221 R        JMP DELNXT
0198                 *
0199   018F EE 27     DEL5    LDX FCBIND,X          POINT TO DIRECTORY ENTRY
0200   0191 A6 00             LDA A 0,X             CHECK FIRST CHARACTER
0201   0193 81 20             CMP A #$20            BLANK? (ALREADY DELETED)
0202   0195 27 11             BEQ DEL5A             YES
0203                 *
0204                         PSHX
0205 +  0197 3F               SWI
0206 +  0198 05               FCB 5
0207   0199 CE 00AC R         LDX #TEMP
0208                         PSHX                  POINT TO FILE NAME
0209 +  019C 3F               SWI
0210 +  019D 05               FCB 5
0211   019E C6 0C             LDA B #12             12 CHARACTER COMPARE
0212                         CMWC                  WILD-CARD COMPARISON
0213 +  01A0 3F               SWI
0214 +  01A1 35               FCB 53
0215   01A2 31                INS
0216   01A3 31                INS                   CLEAN STACK
0217   01A4 31                INS
0218   01A5 31                INS
0219   01A6 27 07             BEQ DEL6              FOUND FILE?
0220                 *
0221   01A8 CE 0000 R DEL5A   LDX #SYSFCB
0222                         GETDR                 GET A NEW DIRECTORY ENTRY
0223 +  01AB 3F               SWI
0224 +  01AC 1A               FCB 26
0225   01AD 20 B4             BRA DEL4A
0226                 *
0227   01AF CE 0000 R DEL6    LDX #SYSFCB
0228   01B2 6C 29             INC FCBSCF,X          MARK 'FILE FOUND'
0229   01B4 CE 022F R         LDX #DPRMPT           OUTPUT 'DELETE-'
0230                         PRTMSG
0231 +  01B7 3F               SWI
0232 +  01B8 31               FCB 49
0233   01B9 CE 0000 R         LDX #SYSFCB
0234   01BC A6 09             LDA A FCBDRV,X        GET DRIVE NUMBER
0235   01BE 8B 30             ADD A #$30            MAKE ASCII
0236   01C0 B7 0239 R         STA A DRIVE
0237   01C3 CE 0239 R         LDX #DRIVE
0238                         PRTMSG                OUTPUT 'DRIVE:'
0239 +  01C6 3F               SWI
0240 +  01C7 31               FCB 49
0241   01C8 CE 0000 R         LDX #SYSFCB
0242   01CB EE 27             LDX FCBIND,X          POINT TO FILE NAME IN DIRECTORY
0243   01CD 86 04             LDA A #$04

0244   01CF A7 0C             STA A 12,X            PUT IN TERMINATOR
0245                         PRTMSG                OUTPUT 'FILENAME.EXT'
0246 +  01D1 3F               SWI
0247 +  01D2 31               FCB 49
0248   01D3 CE 023C R         LDX #QMRK             OUTPUT ' ?'
0249                         PRTMSG
0250 +  01D6 3F               SWI
0251 +  01D7 31               FCB 49
0252                         GTCMD                 GET USER RESPONSE
0253 +  01D8 3F               SWI
0254 +  01D9 30               FCB 48
0255   01DA DE 20             LDX DESCRA
0256   01DC A6 00             LDA A 0,X
0257   01DE 81 59             CMP A #'Y             'YES'?
0258   01E0 26 C6             BNE DEL5A             NO, DO NOT DELETE FILE
0259                 *
0260   01E2 CE 0010 R         LDX #SYSFCB+FCBNAM    POINT TO FCB NAME
0261                         PSHX
0262 +  01E5 3F               SWI
0263 +  01E6 05               FCB 5
0264   01E7 CE 0000 R         LDX #SYSFCB
0265   01EA EE 27             LDX FCBIND,X          POINT TO DIRECTORY NAME
0266   01EC FF 00AA R         STX SAVEX
0267                         PSHX
0268 +  01EF 3F               SWI
0269 +  01F0 05               FCB 5
0270   01F1 C6 0C             LDA B #12             12-CHARACTER MOVE
0271                         MOVC                  MOVE DIR. NAME TO FCB
0272 +  01F3 3F               SWI
0273 +  01F4 11               FCB 17
0274   01F5 31                INS
0275   01F6 31                INS                   CLEAN STACK
0276   01F7 31                INS
0277   01F8 31                INS
0278   01F9 CE 0000 R         LDX #SYSFCB
0279                         DELETE                CALL FILE-DELETE
0280 +  01FC 3F               SWI
0281 +  01FD 1C               FCB 28
0282   01FE B6 00AA R         LDA A SAVEX
0283   0201 F6 00AB R         LDA B SAVEX+1
0284   0204 A7 27             STA A FCBIND,X        RESTORE INDEX
0285   0206 E7 28             STA B FCBIND+1,X
0286   0208 6D 05             TST FCBSTA,X          CHECK STATUS
0287   020A 26 9C             BNE DEL5A             GOOD DELETE?
0288                 *
0289   020C CE 0213 R         LDX #GOOD             YES
0290                         PRTMSG                OUTPUT 'FILE DELETED'
0291 +  020F 3F               SWI
0292 +  0210 31               FCB 49
0293   0211 20 95             BRA DEL5A
0294                 *
0295   0213 20        GOOD    FCC ' FILE DELETED'
0296   0220 0D                FCB $0D
0297                 *
0298   0221 CE 022F R DELNXT  LDX #DPRMPT
0299                         PRTMSG                OUTPUT 'DELETE-'
0300 +  0224 3F               SWI
0301 +  0225 31               FCB 49
0302                         GTCMD                 GET NEW FILE NAME
0303 +  0226 3F               SWI
0304 +  0227 30               FCB 48
```

```
0305  0228 DE 20           LDX DESCRA
0306  022A DF 23           STX CUCHAR          BACK UP ONE TOKEN
0307  022C 7E 00B8 R       JMP DEL0            LOOP AGAIN
0308                 *
0309  022F 20        DPRMPT FCC ' DELETE- '
0310  0238 04              FCB $4
0311                 *
0312  0239 0001      DRIVE  RMB 1
0313  023A 3A              FCC ':'
0314  023B 04              FCB $4
0315                 *
0316  023C 20        QMRK   FCC ' ? '
0317  023F 04              FCB $4
0318                 *
0319                        END
```

```
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
BUFFER 002A R
CHAIN  243A M
CLASS  0026
CLOSE  2369 M
CMPC   231B M
CMWC   2572 M
CUCHAR 0023
DEL0   00B8 R
DEL1   00CA R
DEL1A  00E2 R
DEL1B  00F8 R
DEL2   0104 R
DEL2A  010C R
DEL3   0122 R
DEL4   0143 R
DEL4A  0163 R
DEL4B  018A R
DEL5   018F R
DEL5A  01A8 R
DEL6   01AF R
DELETE 2420 M
DELFIL 0000 RN
DELNXT 0221 R
DESCRA 0020
DESCRC 0022
DIV16  2524 M
DPRMPT 022F R
DRIVE  0239 R
ESCAPE 0043
FCBACS 001E
FCBBAK 000E
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBDTT 0006
FCBEQT 0000
FCBFTS 001F
FCBFWD 000C
FCBGDT 0002
FCBIND 0027
FCBLTS 0021
FCBNAM 0010
FCBNFB 0025
FCBNMS 0023
FCBSCF 0029
FCBSCT 000B
FCBSTA 0005
FCBTRK 000A
FCBTYP 001D
FIBACS 000E
FIBDEF 2940 M
FIBFTS 000F
FIBLTS 0011
FIBNAM 0000
FIBNMS 0013
FIBTYP 000D
FMTFCB 2488 M
FMTS   2558 M
FNFND  017A R
FORMAT 0114 R
GETDR  23EC M
GOOD   0213 R
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
IOHDR  2335 M
LOADB  246E M
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NUMBER 00EA R
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
PRTERR 2454 M
PRTMSG 250A M
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
QMRK   023C R
RC     0025
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
SAVEX  00AA R
SECSIZ 0080
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
SYSFCB 0000 R
TABX   219C M
TEMP   00AC R
TXAB   2183 M
VALUE  0027
WRITE  23D2 M
XABX   21B5 M
```

```
0001   0000 0000    N         NAM INITER
0002                 *
0003                 * INITIALIZE A DISK FOR CP-68 OPERATING SYSTEM
0004                 *
0005                 * FOR ICOM 8 INCH FLOPPY DISKS
0006                 *
0007                 * TRACK 0, SECTORS 1,2       BOOTSTRAP
0008                 * TRACK 0, SECTOR  3         HEADER OF FREE-SPACE LIST
0009                 * TRACK 0, SECTORS 4-26      DIRECTORY SPACE
0010                 * TRACKS 1-77                FREE-SPACE
0011                 *
0012                 * DISK ATTRIBUTES
0013                 *
0014   0000 0080     SECSIZ EQU 128         128 BYTES PER SECTOR
0015   0000 001A     TRKSIZ EQU 26          26 SECTORS PER TRACK
0016   0000 004C     DSKSIZ EQU 76          76 TRACKS ON DISK (LESS TRACK 0)
0017                 *
0018                 * FILE-CONTROL BLOCK ADDRESSES
0019                 *
0020   0000 0005     FCBSTA EQU 5           ERROR STATUS FLAG
0021   0000 0007     FCBDBA EQU 7           DATA BUFFER ADDRESS
0022   0000 0009     FCBDRV EQU 9           DRIVE NUMBER
0023   0000 000A     FCBTRK EQU 10          TRACK NUMBER
0024   0000 000B     FCBSCT EQU 11          SECTOR NUMBER
0025   0000 000C     FCBTLK EQU 12          TRACK LINK POINTER
0026   0000 000D     FCBSLK EQU 13          SECTOR LINK POINTER
0027                 *
0028   0000 0002     FCBSPC RMB 2           FILE-CONTROL BLOCK
0029   0002 44              FCC 'DSK'      DISK
0030   0005 0001            RMB 1
0031   0006 FF              FCB $FF        OUTPUT
0032   0007 0023            RMB 35
0033                 *
0034   002A 0080     BUFFER RMB SECSIZ     SECTOR BUFFER
0035                 *
0036                 * COMMAND-LINE INTERPRETER BASE-PAGE LOCATIONS
0037                 *
0038   00AA 0020     DESCRA EQU $20         ADDRESS OF TOKEN
0039   00AA 0027     VALUE  EQU $27         VALUE OF NUMERIC TOKEN
0040                 *
0041   00AA 49       PROMPT FCC 'INIT. DISK IN DRIVE '
0042   00BE 0001     DRVNO  RMB 1
0043   00BF 20              FCC ' ? '
0044   00C2 04              FCB $04
0045                 *
0046   00C3 00C3    N       ENT . INITR     ENTRY POINT FROM CLI
0047                 *
0048   00C3 96 28     INITR LDA A VALUE+1 GET DRIVE NUMBER
0049   00C5 84 03           AND A #$03     LIMIT RANGE (ICOM PERMITS 4 DRIVES)
0050   00C7 CE 0000 R       LDX #FCBSPC    POINT TO FCB
0051   00CA A7 09           STA A FCBDRV,X
0052   00CC 8B 30           ADD A #$30     MAKE DRIVE NUMBER ASCII
0053   00CE B7 00BE R       STA A DRVNO    PUT IN PROMPT LINE
0054   00D1 CE 00AA R       LDX #PROMPT
0055                        PRTMSG         OUTPUT PROMPT
0056 +  00D4 3F              SWI
0057 +  00D5 31              FCB 49
0058                        GTCMD          GET USER RESPONSE
0059 +  00D6 3F              SWI
0060 +  00D7 30              FCB 48
0061   00D8 DE 20           LDX DESCRA
0062   00DA A6 00           LDA A 0,X      GET FIRST CHAR. OF RESPONSE
0063   00DC 81 59           CMP A #'Y      WAS IT 'YES'?
0064   00DE 27 01           BEQ INITR2     IF SO, CONTINUE
0065                 *
0066   00E0 39              RTS            IF NOT, QUIT
0067                 *
0068   00E1 CE 0000 R INITR2 LDX #FCBSPC    POINT TO FCB
0069   00E4 6F 0A           CLR FCBTRK,X   TRACK=0
0070   00E6 86 01           LDA A #1
0071   00E8 A7 0B           STA A FCBSCT,X  SECTOR=1
0072                        TXAB
0073 +  00EA 3F              SWI
0074 +  00EB 02              FCB 2
0075   00EC CE 0243 R       LDX #BOOT      POINT TO BOOT-CODE
0076                        XABX
0077 +  00EF 3F              SWI
0078 +  00F0 04              FCB 4
0079   00F1 A7 07           STA A FCBDBA,X
0080   00F3 E7 08           STA B FCBDBA+1,X
0081                 *
0082                 * INITIALIZE BOOT BLOCK SECTORS
0083                 *
0084   00F5 8D 46           BSR @WRTBL     WRITE FIRST BLOCK
0085   00F7 6D 05           TST FCBSTA,X   CHECK FOR DISK ERROR
0086   00F9 27 02           BEQ *+4        OK
0087                 *
0088   00FB 20 13           BRA INITQO     FATAL ERROR, QUIT
0089                 *
0090   00FD 6C 0B           INC FCBSCT,X   SECTOR=2
0091                        TXAB
0092 +  00FF 3F              SWI
0093 +  0100 02              FCB 2
0094   0101 CE 02C3 R       LDX #BOOT+SECSIZ
0095                        XABX
0096 +  0104 3F              SWI
0097 +  0105 04              FCB 4
0098   0106 A7 07           STA A FCBDBA,X
0099   0108 E7 08           STA B FCBDBA+1,X
0100   010A 8D 31           BSR @WRTBL     WRITE SECOND BLOCK
0101   010C 6D 05           TST FCBSTA,X   CHECK FOR DISK ERROR
0102   010E 27 02           BEQ *+4        OK
0103                 *
0104   0110 20 7A     INITQO BRA INITQ     FATAL DISK ERROR, QUIT
0105                 *
0106                 *
0107                 * INITIALIZE HEAD OF FREE-SPACE BLOCK
0108                 *
0109                 * ALL ZERO EXCEPT FOR LAST TWO BYTES=TRACK 1, SECTOR 1
0110                 *
0111   0112 6C 0B           INC FCBSCT,X   SECTOR=3
0112                        TXAB
0113 +  0114 3F              SWI
0114 +  0115 02              FCB 2
0115   0116 CE 002A R       LDX #BUFFER
0116                        XABX
0117 +  0119 3F              SWI
0118 +  011A 04              FCB 4
0119   011B A7 07           STA A FCBDBA,X
0120   011D E7 08           STA B FCBDBA+1,X
0121                        PSHX
```

```
0122 +  011F 3F                SWI
0123 +  0120 05                FCB 5
0124                    *
0125                    * CLEAR OUT BUFFER EXCEPT FOR LAST 2 BYTES
0126                    *
0127    0121 CE 002A R          LDX #BUFFER
0128    0124 C6 7E              LDA B #SECSIZ-2
0129    0126 4F                 CLR A
0130    0127 A7 00      INITR3  STA A 0,X
0131    0129 08                 INX
0132    012A 5A                 DEC B
0133    012B 26 FA              BNE INITR3
0134                    *
0135    012D 86 01              LDA A #1        TRACK, SECTOR=1
0136    012F A7 00              STA A 0,X
0137    0131 A7 01              STA A 1,X
0138                            PULX
0139 +  0133 3F                 SWI
0140 +  0134 06                 FCB 6
0141    0135 8D 7E              BSR WRTBLK      WRITE BLOCK 3
0142    0137 6D 05              TST FCBSTA,X    CHECK FOR DISK ERROR
0143    0139 27 04              BEQ *+6         OK
0144                    *
0145    013B 20 4F              BRA INITQ       FATAL DISK ERROR, QUIT
0146                    *
0147    013D 20 76      @WRTBL  BRA WRTBLK      OUT OF RANGE "BSR WRTBLK"
0148    013F 6C 0B              INC FCBSCT,X    SECTOR=4
0149    0141 7F 00A8 R          CLR BUFFER+SECSIZ-2
0150    0144 7F 00A9 R          CLR BUFFER+SECSIZ-1
0151                    *
0152                    * INITIALIZE DIRECTORY TO ZERO
0153                    *
0154    0147 8D 6C      INITR4  BSR WRTBLK      WRITE DIRECTORY BLOCK
0155    0149 6D 05              TST FCBSTA,X    CHECK FOR DISK ERROR
0156    014B 27 02              BEQ *+4         OK
0157                    *
0158    014D 20 3D              BRA INITQ       FATAL DISK ERROR, QUIT
0159                    *
0160    014F A6 0B              LDA A FCBSCT,X
0161    0151 4C                 INC A           NEXT SECTOR
0162    0152 81 1A              CMP A #TRKSIZ   DONE WITH TRACK?
0163    0154 27 04              BEQ INITR5      YES
0164                    *
0165    0156 A7 0B              STA A FCBSCT,X
0166    0158 20 ED              BRA INITR4      NO, CONTINUE WRITING
0167                    *
0168    015A 86 01      INITR5  LDA A #1
0169    015C A7 0B              STA A FCBSCT,X  SECTOR=1
0170    015E A7 0A              STA A FCBTRK,X  TRACK=1
0171    0160 16                 TAB
0172                    *
0173                    * INITIALIZE REST OF DISK (FREE-SPACE)
0174                    *
0175                    *       X=FCB ADDRESS
0176                    *       A=TRACK NUMBER
0177                    *       B=SECTOR NUMBER
0178                    *
0179    0161 5C         INITR6  INC B           MAKE SECTOR LINKAGE
0180    0162 C1 1B              CMP B #TRKSIZ+1 END OF TRACK?
0181    0164 26 09              BNE INITR7      NO
0182                    *
0183    0166 C6 01              LDA B #1        YES, SECTOR=1
0184    0168 4C                 INC A           NEXT TRACK
0185    0169 81 4D              CMP A #DSKSIZ+1 END OF DISK?
0186    016B 26 02              BNE INITR7      NO
0187                    *
0188    016D 4F                 CLR A           LAST SECTOR POINTS TO 0,0
0189    016E 5F                 CLR B
0190                    *
0191    016F B7 002A R  INITR7  STA A BUFFER    TRACK LINK
0192    0172 37                 PSH B           SAVE LSEC
0193    0173 8D 33              BSR GETSC       GET PSEC
0194    0175 F7 002B R          STA B BUFFER+1  SECTOR LINK
0195    0178 33                 PUL B           RESTORE LSEC
0196    0179 8D 3A              BSR WRTBLK      WRITE SECTOR
0197    017B 4D                 TST A           DONE? (=0)
0198    017C 26 04              BNE INITR8      NO
0199                    *
0200    017E 5D                 TST B           DONE? (=0)
0201    017F 26 01              BNE INITR8      NO
0202                    *
0203    0181 39                 RTS             YES, DONE!!!
0204                    *
0205    0182 A7 0A      INITR8  STA A FCBTRK,X
0206    0184 37                 PSH B           SAVE LSEC
0207    0185 8D 21              BSR GETSC       GET PSEC
0208    0187 E7 0B              STA B FCBSCT,X
0209    0189 33                 PUL B           GET LSEC
0210    018A 20 D5              BRA INITR6      KEEP WRITING
0211                    *
0212                    *
0213                    * FATAL ERROR MESSAGE
0214                    *
0215    018C CE 0192 R  INITQ   LDX #QMSG       OUTPUT ERROR MESSAGE
0216                            PRTMSG
0217 +  018F 3F                 SWI
0218 +  0190 31                 FCB 49
0219    0191 39                 RTS             RETURN TO CLI
0220                    *
0221    0192 49         QMSG    FCC 'INITIALIZATION FAILED'
0222    01A7 0D                 FCB $0D
0223                    *
0224                    * CONVERT LSEC TO PSEC
0225                    * LSEC IN B-REG
0226                    *
0227                    GETSC   PSHX            SAVE X-REGISTER
0228 +  01A8 3F                 SWI
0229 +  01A9 05                 FCB 5
0230    01AA CE 0229 R          LDX #TBL        POINT TO LOGICAL/PHYSICAL TABLE
0231                            ADDBX           ADD LOGICAL OFFSET
0232 +  01AD 3F                 SWI
0233 +  01AE 0A                 FCB 10
0234    01AF 09                 DEX             SECTOR STARTS AT 1
0235    01B0 E6 00              LDA B 0,X       GET PSEC
0236                            PULX            RESTORE X-REG
0237 +  01B2 3F                 SWI
0238 +  01B3 06                 FCB 6
0239    01B4 39                 RTS
0240                    *
0241                    * WRITE A SECTOR WITH ERROR CHECKING
0242                    *
0243    01B5 36         WRTBLK  PSH A           SAVE 'A'
```

```
0244   01B6 4F              CLR A
0245   01B7 6F 05           CLR FCBSTA,X  CLEAR ERROR FLAG
0246                        IOHDR         ISSUE I/O REQUEST
0247 +  01B9 3F              SWI
0248 +  01BA 13              FCB 19
0249   01BB A7 05           STA A FCBSTA,X
0250   01BD 4D              TST A         ERROR?
0251   01BE 26 02           BNE WRTERR    YES
0252                *
0253   01C0 32              PUL A         RESTORE 'A'
0254   01C1 39              RTS
0255                *
0256   01C2 16      WRTERR  TAB
0257   01C3 8D 54           BSR OUTHL     CONVERT LEFT DIGIT
0258   01C5 B7 01FF R       STA A ERTYPE
0259   01C8 17              TBA
0260   01C9 8D 52           BSR OUTHR     CONVERT RIGHT DIGIT
0261   01CB B7 0200 R       STA A ERTYPE+1
0262                        PSHX          SAVE X
0263 +  01CE 3F              SWI
0264 +  01CF 05              FCB 5
0265   01D0 A6 0B           LDA A FCBSCT,X
0266   01D2 8D 45           BSR OUTHL     MAKE SECTOR NO.  HEX
0267   01D4 B7 020C R       STA A SECT
0268   01D7 A6 0B           LDA A FCBSCT,X
0269   01D9 8D 42           BSR OUTHR
0270   01DB B7 020D R       STA A SECT+1
0271   01DE A6 0A           LDA A FCBTRK,X
0272   01E0 8D 37           BSR OUTHL     MAKE TRACK NO.  HEX
0273   01E2 B7 0216 R       STA A TRACK
0274   01E5 A6 0A           LDA A FCBTRK,X
0275   01E7 8D 34           BSR OUTHR
0276   01E9 B7 0217 R       STA A TRACK+1
0277   01EC CE 01F4 R       LDX #DERROR
0278                        PRTMSG        PRINT ERROR MESSAGE
0279 +  01EF 3F              SWI
0280 +  01F0 31              FCB 49
0281   01F1 3F              SWI           CALL CP/68
0282   01F2 1F              FCB 31        "WARMSTART"
0283   01F3 39              RTS
0284                *
0285   01F4 44      DERROR  FCC 'DISK ERROR:'
0286   01FF 0002    ERTYPE  RMB 2
0287   0201 20              FCC ' AT SECTOR '
0288   020C 0002    SECT    RMB 2
0289   020E 2C              FCC ', TRACK '
0290   0216 0002    TRACK   RMB 2
0291   0218 0D              FCB $0D
0292                *
0293                * CONVERT BINARY TO HEX-ASCII HERE
0294                *
0295   0219 44      OUTHL   LSR A         SHIFT RIGHT
0296   021A 44              LSR A
0297   021B 44              LSR A
0298   021C 44              LSR A
0299                *
0300   021D 84 0F   OUTHR   AND A #$0F    GET NIBBLE
0301   021F 8B 30           ADD A #$30    MAKE ASCII
0302   0221 81 39           CMP A #$39    >9?
0303   0223 23 02           BLS *+4       NO
0304                *
0305   0225 8B 07           ADD A #$7     YES
0306                *
0307   0227 39              RTS
0308                *
0309                *
0310                * LOGICAL/PHYSICAL SECTOR TABLE
0311                *
0312   0228 00              FCB 00
0313                *
0314   0229 01      TBL     FCB $1
0315   022A 0A              FCB $A
0316   022B 13              FCB $13
0317   022C 02              FCB $2
0318   022D 0B              FCB $B
0319   022E 14              FCB $14
0320   022F 03              FCB $3
0321   0230 0C              FCB $C
0322   0231 15              FCB $15
0323   0232 04              FCB $4
0324   0233 0D              FCB $D
0325   0234 16              FCB $16
0326   0235 05              FCB $5
0327   0236 0E              FCB $E
0328   0237 17              FCB $17
0329   0238 06              FCB $6
0330   0239 0F              FCB $F
0331   023A 18              FCB $18
0332   023B 07              FCB $7
0333   023C 10              FCB $10
0334   023D 19              FCB $19
0335   023E 08              FCB $8
0336   023F 11              FCB $11
0337   0240 1A              FCB $1A
0338   0241 09              FCB $9
0339   0242 12              FCB $12
0340                *
0341                *
0342                *
0343   0243 0243    R BOOT  EQU *         . BOOT PROGRAM STARTS HERE
0344                *
0345                *
0346                        END
```

```
 INITR 00C3 RN
@WRTBL 013D R
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
BOOT   0243 R
BUFFER 002A R
CHAIN  243A M
CLOSE  2369 M
CMPC   231B M
CMWC   2572 M
DELETE 2420 M
DERROR 01F4 R
DESCRA 0020
DIV16  2524 M
DRVNO  00BE R
DSKSIZ 004C
ERTYPE 01FF R
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBSCT 000B
FCBSLK 000D
FCBSPC 0000 R
FCBSTA 0005
FCBTLK 000C
FCBTRK 000A
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
GETDR  23EC M
GETSC  01A8 R
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
INITER 0000 RN
INITQ  018C R
INITQ0 0110 R
INITR2 00E1 R
INITR3 0127 R
INITR4 0147 R
INITR5 015A R
INITR6 0161 R
INITR7 016F R
INITR8 0182 R
IOHDR  2335 M
LOADB  246E M
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
OUTHL  0219 R
OUTHR  021D R
PROMPT 00AA R
PRTERR 2454 M
PRTMSG 250A M
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
QMSG   0192 R
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
SECSIZ 0080
SECT   020C R
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
TABX   219C M
TBL    0229 R
TRACK  0216 R
TRKSIZ 001A
TXAB   2183 M
VALUE  0027
WRITE  23D2 M
WRTBLK 01B5 R
WRTERR 01C2 R
XABX   21B5 M
```

```
0001    0000 0000    N       NAM LINKER
0002                         *
0003                         * TRANSIENT COMMAND 'LINK' PROCESSOR
0004                         * SYNTAX:  LINK [DRIVE:] FILENAME.EXT
0005                         * MAKE SYSTEM LINKAGE TO FILENAME.EXT
0006                         *
0007                         * BLOCK ADDRESSING DEFINITIONS
0008                         *
0009                               FCBDEF
0010 +  0000 0000             FCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0011 +  0000 0002             FCBGDT EQU 2          GENERIC DEVICE TYPE
0012 +  0000 0005             FCBSTA EQU 5          STATUS
0013 +  0000 0006             FCBDTT EQU 6          DATA TRANSFER TYPE
0014 +  0000 0007             FCBDBA EQU 7          DATA BUFFER ADDRESS
0015 +  0000 0009             FCBDRV EQU 9          DRIVE NUMBER
0016 +  0000 000A             FCBTRK EQU 10         TRACK NUMBER
0017 +  0000 000B             FCBSCT EQU 11         SECTOR NUMBER
0018 +  0000 000C             FCBFWD EQU 12         FWD LINK TRACK/SECTOR
0019 +  0000 000E             FCBBAK EQU 14         BACK LINK TRACK/SECTOR
0020 +  0000 0010             FCBNAM EQU 16         FILE NAME (8.3+EOT=13))
0021 +  0000 001D             FCBTYP EQU 29         FILE TYPE
0022 +  0000 001E             FCBACS EQU 30         FILE ACCESS CODE
0023 +  0000 001F             FCBFTS EQU 31         FIRST TRACK/SECTOR
0024 +  0000 0021             FCBLTS EQU 33         LAST TRACK/SECTOR
0025 +  0000 0023             FCBNMS EQU 35         NUMBER OF SECTORS
0026 +  0000 0025             FCBNFB EQU 37         NEXT FCB IN ACTIVE CHAIN
0027 +  0000 0027             FCBIND EQU 39         INDEX INTO DATA BUFFER
0028 +  0000 0029             FCBSCF EQU 41         SPACE COMPRESSION FLAG
0029                               FIBDEF
0030 +  0000 0000             FIBNAM EQU 0          FILE NAME (8.3 + EOT=13)
0031 +  0000 000D             FIBTYP EQU 13         FILE TYPE
0032 +  0000 000E             FIBACS EQU 14         FILE ACCESS CODE
0033 +  0000 000F             FIBFTS EQU 15         FIRST TRACK/SECTOR
0034 +  0000 0011             FIBLTS EQU 17         LAST TRACK/SECTOR
0035 +  0000 0013             FIBNMS EQU 19         NUMBER OF SECTORS
0036                         *
0037                         * BASE-PAGE EQUATES
0038                         *
0039    0000 0020            DESCRA EQU   $20
0040    0000 0022            DESCRC EQU   $22
0041    0000 0023            CUCHAR EQU   $23
0042    0000 0025            RC     EQU   $25
0043    0000 0026            CLASS  EQU   $26
0044    0000 0027            VALUE  EQU   $27
0045                         *
0046                         * DISK ATTRIBUTES
0047                         *
0048    0000 0080            SECSIZ EQU   128          128 BYTES/SECTOR
0049                         *
0050                         * FCB FOR TRANSIENT
0051                         *
0052    0000 0002            SYSFCB RMB 2
0053    0002 44                     FCC 'DSK'
0054    0005 0002                   RMB 2
0055    0007 002A    R              FDB BUFFER
0056    0009 0021                   RMB 33
0057    002A 0080            BUFFER RMB SECSIZ
0058                         *
0059    00AA CE 0000 R LINK  LDX #SYSFCB
0060    00AD 6F 09                  CLR FCBDRV,X      DEFAULT DRIVE=0
```

```
0061   00AF 6F 06             CLR FCBDTT,X        INPUT
0062   00B1 CE 01CE R         LDX #PRMPT          ISSUE OPERATOR PROMPT
0063                          PRTMSG
0064 +  00B4 3F                SWI
0065 +  00B5 31                FCB 49
0066                          GTCMD               GET USER RESPONSE
0067 +  00B6 3F                SWI
0068 +  00B7 30                FCB 48
0069   00B8 CE 0000 R         LDX #SYSFCB
0070   00BB D6 25             LDA B RC            CHECK RC
0071   00BD C1 03             CMP B #3            NUMBER?
0072   00BF 26 2F             BNE LNK2            NO
0073                 *
0074   00C1 7D 0027           TST VALUE           VALID DRIVE NO.?
0075   00C4 26 0A             BNE LNK1            NO, ERROR
0076                 *
0077   00C6 96 28             LDA A VALUE+1       VALID DRIVE NO.?
0078   00C8 81 03             CMP A #3            (4 DRIVES)
0079   00CA 22 04             BHI LNK1            NO
0080                 *
0081   00CC A7 09             STA A FCBDRV,X      SET DRIVE NO.
0082   00CE 20 14             BRA LNK1A
0083                 *
0084   00D0 CE 00D6 R LNK1    LDX #NUMBER         NUMBER ERROR
0085                          PRTMSG
0086 +  00D3 3F                SWI
0087 +  00D4 31                FCB 49
0088   00D5 39                RTS
0089                 *
0090   00D6 20        NUMBER  FCC ' NUMBER ERROR'
0091   00E3 0D                FCB $0D
0092                 *
0093                 LNK1A     NXTOK             GET TOKEN FROM CLI
0094 +  00E4 3F                SWI
0095 +  00E5 2F                FCB 47
0096   00E6 D6 25             LDA B RC            CHECK RC
0097   00E8 C1 3A             CMP B #':           COLON?
0098   00EA 26 E4             BNE LNK1            IF NOT, ERROR
0099                 *
0100                          NXTOK               GET TOKEN FROM CLI
0101 +  00EC 3F                SWI
0102 +  00ED 2F                FCB 47
0103   00EE D6 25             LDA B RC            CHECK RC
0104   00F0 C1 01     LNK2    CMP B #1            UNAMBIG. NAME?
0105   00F2 27 14             BEQ LNK4            YES
0106                 *
0107   00F4 CE 00FA R LNK3    LDX #FORMAT         FORMAT ERROR
0108                          PRTMSG
0109 +  00F7 3F                SWI
0110 +  00F8 31                FCB 49
0111   00F9 39                RTS
0112                 *
0113   00FA 20        FORMAT  FCC ' FORMAT ERROR'
0114   0107 0D                FCB $0D
0115                 *
0116   0108 DE 20     LNK4    LDX DESCRA          POINT TO NAME
0117   010A FF 002A R         STX BUFFER
0118   010D 96 22             LDA A DESCRC        GET LENGTH OF NAME
0119   010F B7 002C R         STA A BUFFER+2
0120                          NXTOK               GET TOKEN FROM CLI
0121 +  0112 3F                SWI
0122 +  0113 2F                FCB 47
0123   0114 D6 25             LDA B RC            CHECK RC
0124   0116 C1 2E             CMP B #'            PERIOD?
0125   0118 26 DA             BNE LNK3            IF NOT, ERROR
0126                 *
0127   011A 7C 002C R         INC BUFFER+2        COUNT PERIOD
0128                          NXTOK               GET TOKEN FROM CLI
0129 +  011D 3F                SWI
0130 +  011E 2F                FCB 47
0131   011F D6 25             LDA B RC            CHECK RC
0132   0121 C1 01             CMP B #1            UNAMBIG. NAME?
0133   0123 26 CF             BNE LNK3            IF NOT, ERROR
0134                 *
0135   0125 D6 22             LDA B DESCRC        GET LENGTH OF EXT
0136   0127 FB 002C R         ADD B BUFFER+2      TOTAL LENGTH
0137   012A CE 0010 R         LDX #SYSFCB+FCBNAM
0138                          PSHX                POINTER TO FCBNAM
0139 +  012D 3F                SWI
0140 +  012E 05                FCB 5
0141   012F FE 002A R         LDX BUFFER
0142                          PSHX                POINTER TO CLI NAME
0143 +  0132 3F                SWI
0144 +  0133 05                FCB 5
0145                          FMTS                FORMAT NAME INTO FCB
0146 +  0134 3F                SWI
0147 +  0135 34                FCB 52
0148   0136 31                INS
0149   0137 31                INS                 CLEAN STACK
0150   0138 31                INS
0151   0139 31                INS
0152   013A 5D                TST B               ERRORS?
0153   013B 26 B7             BNE LNK3            YES
0154                 *
0155   013D CE 0000 R         LDX #SYSFCB
0156                          OPEND               OPEN THE DIRECTORY
0157 +  0140 3F                SWI
0158 +  0141 17                FCB 23
0159   0142 A6 05     LNK5    LDA A FCBSTA,X      CHECK STATUS
0160   0144 27 1D             BEQ LNK6            GOOD?
0161                 *
0162   0146 81 01             CMP A #1            END OF DIRECTORY?
0163   0148 26 16             BNE LNK5A           NO
0164                 *
0165   014A CE 0150 R         LDX #FNFND          FILE NOT FOUND ON DISK
0166                          PRTMSG
0167 +  014D 3F                SWI
0168 +  014E 31                FCB 49
0169   014F 39                RTS
0170                 *
0171   0150 20        FNFND   FCC ' FILE NOT FOUND'
0172   015F 0D                FCB $0D
0173                 *
0174                 LNK5A     PRTERR            PRINT ERROR MESSAGE
0175 +  0160 3F                SWI
0176 +  0161 1E                FCB 30
0177   0162 39                RTS
0178                 *
0179   0163 EE 27     LNK6    LDX FCBIND,X        POINT TO DIRECTORY NAME
0180                          PSHX
0181 +  0165 3F                SWI
0182 +  0166 05                FCB 5
```

```
0183   0167 CE 0010 R        LDX #SYSFCB+FCBNAM   POINT TO FCB NAME
0184                         PSHX
0185 +  016A 3F               SWI
0186 +  016B 05               FCB 5
0187   016C C6 0C            LDA B #12            COMPARE 12 CHARACTERS
0188                         CMPC
0189 +  016E 3F               SWI
0190 +  016F 12               FCB 18
0191   0170 31               INS
0192   0171 31               INS                  CLEAN STACK
0193   0172 31               INS
0194   0173 31               INS
0195   0174 27 07            BEQ LNK7             FOUND ENTRY IN DIRECTORY?
0196                  *
0197   0176 CE 0000 R        LDX #SYSFCB
0198                         GETDR                GET NEW ENTRY
0199 +  0179 3F               SWI
0200 +  017A 1A               FCB 26
0201   017B 20 C5            BRA LNK5
0202                  *
0203   017D CE 0000 R LNK7    LDX #SYSFCB
0204   0180 EE 27            LDX FCBIND,X         POINT TO DIRECTORY ENTRY
0205   0182 A6 0F            LDA A FIBFTS,X       GET FIRST T/S
0206   0184 E6 10            LDA B FIBFTS+1,X
0207   0186 CE 0000 R        LDX #SYSFCB
0208   0189 A7 1F            STA A FCBFTS,X       SAVE IN FCB
0209   018B E7 20            STA B FCBFTS+1,X
0210   018D EE 27            LDX FCBIND,X
0211   018F A6 11            LDA A FIBLTS,X       GET LAST T/S
0212   0191 E6 12            LDA B FIBLTS+1,X
0213   0193 CE 0000 R        LDX #SYSFCB
0214   0196 A7 21            STA A FCBLTS,X       SAVE IN FCB
0215   0198 E7 22            STA B FCBLTS+1,X
0216   019A 86 00            LDA A #0             TRACK=0
0217   019C C6 03            LDA B #3             SECTOR=3
0218   019E A7 0A            STA A FCBTRK,X
0219   01A0 E7 0B            STA B FCBSCT,X       GET LINK SECTOR
0220                         IOHDR
0221 +  01A2 3F               SWI
0222 +  01A3 13               FCB 19
0223   01A4 6D 05            TST FCBSTA,X         ERROR?
0224   01A6 27 03            BEQ *+5
0225                  *
0226   01A8 7E 0160 R        JMP LNK5A            ERROR MESSAGE
0227                  *
0228   01AB CE 0000 R        LDX #SYSFCB
0229   01AE 63 06            COM FCBDTT,X         MAKE 'OUTPUT'
0230   01B0 A6 1F            LDA A FCBFTS,X       GET LINKAGE INFO.
0231   01B2 E6 20            LDA B FCBFTS+1,X
0232   01B4 B7 00A4 R        STA A BUFFER+122     PUT IN LINKAGE SECTOR
0233   01B7 F7 00A5 R        STA B BUFFER+123
0234   01BA A6 21            LDA A FCBLTS,X
0235   01BC E6 22            LDA B FCBLTS+1,X
0236   01BE B7 00A6 R        STA A BUFFER+124
0237   01C1 F7 00A7 R        STA B BUFFER+125
0238                         IOHDR                WRITE LINKAGE SECTOR
0239 +  01C4 3F               SWI
0240 +  01C5 13               FCB 19
0241   01C6 6D 05            TST FCBSTA,X         ERROR?
0242   01C8 27 03            BEQ *+5              NO
0243                  *
0244   01CA 7E 0160 R        JMP LNK5A            YES
0245                  *
0246   01CD 39               RTS
0247                  *
0248                  *
0249   01CE 20        PRMPT  FCC ' SYSTEM FILE NAME? '
0250   01E1 04               FCB $04
0251                  *
0252                         END
```

```
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
BUFFER 002A R
CHAIN  243A M
CLASS  0026
CLOSE  2369 M
CMPC   231B M
CMWC   2572 M
CUCHAR 0023
DELETE 2420 M
DESCRA 0020
DESCRC 0022
DIV16  2524 M
FCBACS 001E
FCBBAK 000E
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBDTT 0006
FCBEQT 0000
FCBFTS 001F
FCBFWD 000C
FCBGDT 0002
FCBIND 0027
FCBLTS 0021
FCBNAM 0010
FCBNFB 0025
FCBNMS 0023
FCBSCF 0029
FCBSCT 000B
FCBSTA 0005
FCBTRK 000A
FCBTYP 001D
FIBACS 000E
FIBDEF 2940 M
FIBFTS 000F
FIBLTS 0011
FIBNAM 0000
FIBNMS 0013
FIBTYP 000D
FMTFCB 2488 M
FMTS   2558 M
FNFND  0150 R
FORMAT 00FA R
GETDR  23EC M
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
IOHDR  2335 M
LINK   00AA R
LINKER 0000 RN
LNK1   00D0 R
LNK1A  00E4 R
LNK2   00F0 R
LNK3   00F4 R
LNK4   0108 R
LNK5   0142 R
LNK5A  0160 R
LNK6   0163 R
LNK7   017D R
LOADB  246E M
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NUMBER 00D6 R
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
PRMPT  01CE R
PRTERR 2454 M
PRTMSG 250A M
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
RC     0025
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
SECSIZ 0080
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
SYSFCB 0000 R
TABX   219C M
TXAB   2183 M
VALUE  0027
WRITE  23D2 M
XABX   21B5 M
```

```
0001    0000 0000    N        NAM PIPPER
0002                          *
0003                          * TRANSIENT PERIPHERAL-INTERCHANGE "PIP"
0004                          *
0005                          *       SET UP ADDRESSING EQUATES
0006                          *
0007                                BASEQU
0008 +  0000 0020       DESCRA EQU $20  DESCRIPTOR ADDRESS(2)
0009 +  0000 0022       DESCRC EQU $22  DESCRIPTOR COUNT
0010 +  0000 0023       CUCHAR EQU $23  CURRENT CHAR (2)
0011 +  0000 0025       RC     EQU $25  TOKEN RETURN CODE
0012 +  0000 0026       CLASS  EQU $26  TOKEN CLASS
0013 +  0000 0027       VALUE  EQU $27  BIN VALUE/TRANSFER ADDRESS (2)
0014 +  0000 0029       FCBCHN EQU $29  TOP OF FCB CHAIN (2)
0015 +  0000 002B       FRETAB EQU $2B  DISK FREE SPACE POINTER (8)
0016 +  0000 0033       BMEM   EQU $33  START OF TRANSIENT AREA(2)
0017 +  0000 0035       EMEM   EQU $35  END OF TRANSIENT AREA (2)
0018 +  0000 0037       CMEM   EQU $37  NEXT AVAIL TRANSIENT AREA (2)
0019 +  0000 0039       BS     EQU $39  BACKSPACE CHAR
0020 +  0000 003A       DL     EQU $3A  DELETE LINE CHAR
0021 +  0000 003B       DP     EQU $3B  DEPTH; LINES/PAGE
0022 +  0000 003C       DPCNT  EQU $3C  DEPTH TEMP
0023 +  0000 003D       WD     EQU $3D  WIDTH; CHARS/LINE
0024 +  0000 003E       NL     EQU $3E  NULL COUNT
0025 +  0000 003F       TB     EQU $3F  TAB CHAR
0026 +  0000 0040       DX     EQU $40  DUPLEX; FF=H, 00=F
0027 +  0000 0041       EJ     EQU $41  EJECT COUNT
0028 +  0000 0042       PS     EQU $42  PAUSE; 00=YES
0029 +  0000 0043       ES     EQU $43  ESCAPE CHAR
0030 +  0000 0044       LDP    EQU $44  DEPTH LINES/PAGE
0031 +  0000 0045       LDPCNT EQU $45  DEPTH TEMP
0032 +  0000 0046       LWD    EQU $46  WIDTH CHARS/LINE
0033                              RCBDEF
0034 +  0000 0000       RCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0035 +  0000 0002       RCBGDT EQU 2          GENERIC DEVICE TYPE
0036 +  0000 0005       RCBSTA EQU 5          STATUS
0037 +  0000 0006       RCBDTT EQU 6          DATA TRANSFER TYPE
0038 +  0000 0007       RCBDBA EQU 7          DATA BUFFER ADDRESS
0039                              FCBDEF
0040 +  0000 0000       FCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0041 +  0000 0002       FCBGDT EQU 2          GENERIC DEVICE TYPE
0042 +  0000 0005       FCBSTA EQU 5          STATUS
0043 +  0000 0006       FCBDTT EQU 6          DATA TRANSFER TYPE
0044 +  0000 0007       FCBDBA EQU 7          DATA BUFFER ADDRESS
0045 +  0000 0009       FCBDRV EQU 9          DRIVE NUMBER
0046 +  0000 000A       FCBTRK EQU 10         TRACK NUMBER
0047 +  0000 000B       FCBSCT EQU 11         SECTOR NUMBER
0048 +  0000 000C       FCBFWD EQU 12         FWD LINK TRACK/SECTOR
0049 +  0000 000E       FCBBAK EQU 14         BACK LINK TRACK/SECTOR
0050 +  0000 0010       FCBNAM EQU 16         FILE NAME (8.3+EOT=13))
0051 +  0000 001D       FCBTYP EQU 29         FILE TYPE
0052 +  0000 001E       FCBACS EQU 30         FILE ACCESS CODE
0053 +  0000 001F       FCBFTS EQU 31         FIRST TRACK/SECTOR
0054 +  0000 0021       FCBLTS EQU 33         LAST TRACK/SECTOR
0055 +  0000 0023       FCBNMS EQU 35         NUMBER OF SECTORS
0056 +  0000 0025       FCBNFB EQU 37         NEXT FCB IN ACTIVE CHAIN
0057 +  0000 0027       FCBIND EQU 39         INDEX INTO DATA BUFFER
0058 +  0000 0029       FCBSCF EQU 41         SPACE COMPRESSION FLAG
0059                              FIBDEF
0060 +  0000 0000       FIBNAM EQU 0          FILE NAME (8.3 + EOT=13)
```

```
0061 +  0000 000D       FIBTYP EQU 13          FILE TYPE
0062 +  0000 000E       FIBACS EQU 14          FILE ACCESS CODE
0063 +  0000 000F       FIBFTS EQU 15          FIRST TRACK/SECTOR
0064 +  0000 0011       FIBLTS EQU 17          LAST TRACK/SECTOR
0065 +  0000 0013       FIBNMS EQU 19          NUMBER OF SECTORS
0066                     *
0067                     * FILE CONTROL BLOCKS FOR "PIP"
0068                     *
0069    0000 0006        INFCB  RMB 6
0070    0006 00                 FCB 0
0071    0007 002C     R         FDB INBUF
0072    0009 0021               RMB 33
0073    002A 0002        INHND  RMB 2           ADDRESS OF INPUT DEVICE IN DEVTAB
0074                     *
0075    002C 0100        INBUF  RMB 256         INPUT BUFFER
0076                     *
0077    012C 0006        OUTFCB RMB 6
0078    0132 FF                 FCB $FF
0079    0133 0158     R         FDB OUTBUF
0080    0135 0021               RMB 33
0081    0156 0002        OUTHND RMB 2           ADDRESS OF OUTPUT DEVICE IN DEVTAB
0082                     *
0083    0158 0100        OUTBUF RMB 256         OUTPUT BUFFER
0084                     *
0085                     *
0086                     * DISK ATTRIBUTES
0087                     *
0088    0258 0000        FSTTRK EQU 0           NUMBER OF FIRST TRACK ON DISK
0089    0258 0001        FSTSEC EQU 1           NUMBER OF FIRST SECTOR ON DISK
0090    0258 0080        SECSIZ EQU 128         NUMBER OF BYTES/SECTOR
0091    0258 001A        TRKSIZ EQU 26          NUMBER OF SECTORS/TRACK
0092    0258 004D        DSKSIZ EQU 77          NUMBER OF TRACKS ON DISK
0093                     *
0094                     *
0095    0258 0001        HFLAG  RMB 1           HEX-FORMAT FLAG
0096    0259 0001        BFLAG  RMB 1           BINARY-FORMAT FLAG
0097                     *
0098                     *
0099                     * DEVICE TABLE
0100                     *
0101                     * FIRST ENTRY:  OPEN HANDLER
0102                     * SECOND:       CLOSE HANDLER
0103                     * THIRD:        READ A CHARACTER
0104                     * FOURTH:       WRITE A CHARACTER
0105                     *
0106    025A 43          DEVTAB FCC 'CON'
0107    025D 02DE     R         FDB LOPEN
0108    025F 0321     R         FDB LCLOSE
0109    0261 0330     R         FDB LREAD
0110    0263 035B     R         FDB LWRITE
0111                       *
0112    0265 50                  FCC 'PTR'
0113    0268 02DE     R         FDB LOPEN
0114    026A 0321     R         FDB LCLOSE
0115    026C 0330     R         FDB LREAD
0116    026E 0000                FDB 0         ILLEGAL OUTPUT DEVICE
0117                       *
0118    0270 50                  FCC 'PTP'
0119    0273 02DE     R         FDB LOPEN
0120    0275 0321     R         FDB LCLOSE
0121    0277 0000                FDB 0         ILLEGAL INPUT DEVICE
0122    0279 035B     R         FDB LWRITE
0123                       *
0124    027B 44          DSKDEV FCC 'DSK'
0125    027E 02B3     R         FDB DOPEN
0126    0280 02BF     R         FDB DCLOSE
0127    0282 02C4     R         FDB DREAD
0128    0284 02D4     R         FDB DWRITE
0129                       *
0130    0286 4C                  FCC 'LPT'
0131    0289 0388     R         FDB SOPEN
0132    028B 038F     R         FDB SCLOSE
0133    028D 0000                FDB 0         ILLEGAL INPUT DEVICE
0134    028F 035B     R         FDB LWRITE
0135                       *
0136    0291 4D                  FCC 'MTA'
0137    0294 02DE     R         FDB LOPEN
0138    0296 0321     R         FDB LCLOSE
0139    0298 0330     R         FDB LREAD
0140    029A 035B     R         FDB LWRITE
0141                       *
0142    029C 54                  FCC 'TTY'
0143    029F 02DE     R         FDB LOPEN
0144    02A1 0321     R         FDB LCLOSE
0145    02A3 0330     R         FDB LREAD
0146    02A5 035B     R         FDB LWRITE
0147                       *
0148    02A7 4E                  FCC 'NUL'
0149    02AA 02DE     R         FDB LOPEN
0150    02AC 0321     R         FDB LCLOSE
0151    02AE 0000                FDB 0         ILLEGAL INPUT DEVICE
0152    02B0 035B     R         FDB LWRITE
0153                       *
0154    02B2 00                  FCB 0         END OF TABLE
0156                     * CHARACTER-ORIENTED I/O HANDLERS
0157                     *
0158                     * D-PREFIX INDICATES THAT THE DEVICE SUPPORTS CHARACTER I/O
0159                     * L-PREFIX INDICATES THAT THE DEVICE SUPPORTS BLOCK I/O
0160                     *
0161                     DOPEN  TABX           POINT TO FCB
0162 +  02B3 3F                 SWI
0163 +  02B4 03                 FCB 3
0164                            OPEN
0165 +  02B5 3F                 SWI
0166 +  02B6 14                 FCB 20
0167    02B7 6D 05               TST FCBSTA,X  CHECK STATUS
0168    02B9 26 01               BNE IOERR     IF BAD, ERROR MESSAGE
0169                       *
0170    02BB 39                  RTS
0171                       *
0172                     IOERR  PRTERR         PRINT ERROR MESSAGE
0173 +  02BC 3F                 SWI
0174 +  02BD 1E                 FCB 30
0175    02BE 39                  RTS
0176                       *
0177                       *
0178                       *
0179                     DCLOSE TABX           POINT TO FCB
0180 +  02BF 3F                 SWI
0181 +  02C0 03                 FCB 3
0182                            CLOSE
0183 +  02C1 3F                 SWI
```

```
0184 +  02C2 15                FCB 21
0185    02C3 39                RTS
0186                    *
0187                    *
0188                    *
0189    02C4 CE 0000 R DREAD   LDX #INFCB
0190                           READ
0191 +  02C7 3F                SWI
0192 +  02C8 18                FCB 24
0193    02C9 E6 05             LDA B FCBSTA,X  CHECK STATUS
0194    02CB C1 08             CMP B #8        EOF?
0195    02CD 26 01             BNE *+3         NO
0196                    *
0197    02CF 39                RTS             YES
0198                    *
0199    02D0 5D                TST B           ERROR?
0200    02D1 26 E9             BNE IOERR       IF BAD, ERROR MESSAGE
0201                    *
0202    02D3 39                RTS
0203                    *
0204                    *
0205                    *
0206    02D4 CE 012C R DWRITE  LDX #OUTFCB
0207                           WRITE
0208 +  02D7 3F                SWI
0209 +  02D8 19                FCB 25
0210    02D9 6D 05             TST FCBSTA,X    CHECK STATUS
0211    02DB 26 DF             BNE IOERR       IF BAD, ERROR MESSAGE
0212                    *
0213    02DD 39                RTS
0214                    *
0215                    *
0216                    *
0217                    LOPEN  TABX            POINT TO FCB
0218 +  02DE 3F                SWI
0219 +  02DF 03                FCB 3
0220    02E0 6D 06             TST FCBDTT,X    INPUT OR OUTPUT?
0221    02E2 26 24             BNE LOPOUT
0222                    *
0223    02E4 FE 002A R         LDX INHND       CHECK FOR VALID INPUT DEVICE
0224    02E7 EE 07             LDX 7,X
0225    02E9 26 0D             BNE LOP2        OK?
0226                    *
0227    02EB CE 03A7 R         LDX #ERR3       NO, ERROR MESSAGE
0228                           PRTMSG
0229 +  02EE 3F                SWI
0230 +  02EF 31                FCB 49
0231    02F0 CE 0000 R         LDX #INFCB
0232    02F3 86 12             LDA A #18       RETURN ERROR STATUS
0233    02F5 A7 05             STA A FCBSTA,X
0234    02F7 39                RTS
0235                    *
0236    02F8 CE 0000 R LOP2    LDX #INFCB
0237                    *
0238                    LOP2A  IOHDR           ISSUE I/O REQUEST
0239 +  02FB 3F                SWI
0240 +  02FC 13                FCB 19
0241                    *
0242    02FD A6 07     LOP2B   LDA A FCBDBA,X
0243    02FF E6 08             LDA B FCBDBA+1,X  RE-INIT. FCBINDEX
0244    0301 A7 27             STA A FCBIND,X
0245    0303 E7 28             STA B FCBIND+1,X
0246    0305 6F 05             CLR FCBSTA,X    RETURN GOOD STATUS
0247    0307 39                RTS
0248                    *
0249    0308 FE 0156 R LOPOUT  LDX OUTHND      CHECK FOR VALID OUTPUT DEVICE
0250    030B EE 09             LDX 9,X
0251    030D 26 0D             BNE LOP3        OK?
0252                    *
0253    030F CE 03BD R         LDX #ERR4       NO, ERROR MESSAGE
0254                           PRTMSG
0255 +  0312 3F                SWI
0256 +  0313 31                FCB 49
0257    0314 CE 012C R         LDX #OUTFCB
0258    0317 86 12             LDA A #18       RETURN ERROR STATUS
0259    0319 A7 05             STA A FCBSTA,X
0260    031B 39                RTS
0261                    *
0262    031C CE 012C R LOP3    LDX #OUTFCB
0263    031F 20 DC             BRA LOP2B
0264                    *
0265                    *
0266                    *
0267                    LCLOSE TABX            POINT TO FCB
0268 +  0321 3F                SWI
0269 +  0322 03                FCB 3
0270    0323 6D 06             TST FCBDTT,X    INPUT OR OUTPUT?
0271    0325 26 01             BNE LCLOS2
0272                    *
0273    0327 39                RTS             INPUT DOES NOTHING
0274                    *
0275    0328 86 04     LCLOS2  LDA A #$04       OUTPUT CNTL-D (EOF)
0276    032A 8D 2F             BSR LWRITE
0277    032C 86 0D             LDA A #$0D      OUTPUT CARRIAGE RETURN
0278    032E 20 2B             BRA LWRITE
0279                    *
0280                    *
0281                    *
0282    0330 CE 0000 R LREAD   LDX #INFCB
0283    0333 EE 27             LDX FCBIND,X    POINT TO BUFFER
0284    0335 A6 00             LDA A 0,X       GET CHARACTER
0285    0337 81 04             CMP A #$04      CNTL-D (EOF)?
0286    0339 26 08             BNE LREAD2      NO
0287                    *
0288    033B CE 0000 R         LDX #INFCB
0289    033E C6 08             LDA B #8        RETURN EOF STATUS
0290    0340 E7 05             STA B FCBSTA,X
0291    0342 39                RTS
0292                    *
0293    0343 81 0D     LREAD2  CMP A #$0D      CARRIAGE-RETURN?
0294    0345 27 0F             BEQ LREAD3      YES
0295                    *
0296    0347 08                INX             MOVE POINTER
0297    0348 36                PSH A           SAVE 'A'
0298                           TXAB
0299 +  0349 3F                SWI
0300 +  034A 02                FCB 2
0301    034B CE 0000 R         LDX #INFCB      POINT TO FCB
0302    034E A7 27             STA A FCBIND,X
0303    0350 E7 28             STA B FCBIND+1,X
0304    0352 32                PUL A           RECOVER 'A'
0305    0353 6F 05             CLR FCBSTA,X    RETURN GOOD STATUS
```

```
0306   0355 39              RTS
0307                *
0308   0356 8D A0    LREAD3 BSR LOP2       INPUT A BLOCK AND RESET INDEX
0309   0358 86 0D           LDA A #$0D     RELOAD CR.
0310   035A 39              RTS
0311                *
0312                *
0313                *
0314   035B CE 012C R LWRITE LDX #OUTFCB
0315   035E EE 27           LDX FCBIND,X   GET BUFFER POINTER
0316   0360 A7 00           STA A 0,X      STORE CHARACTER
0317   0362 08              INX
0318   0363 8C 0258 R       CPX #OUTBUF+256  OUT OF BUFFER?
0319   0366 27 13           BEQ LWRIT2       YES, ERROR
0320                *
0321   0368 36              PSH A          SAVE 'A'
0322                        TXAB
0323 +  0369 3F              SWI
0324 +  036A 02              FCB 2
0325   036B CE 012C R       LDX #OUTFCB    POINT TO FCB
0326   036E A7 27           STA A FCBIND,X
0327   0370 E7 28           STA B FCBIND+1,X
0328   0372 32              PUL A          RECOVER 'A'
0329   0373 CE 012C R       LDX #OUTFCB
0330   0376 81 0D           CMP A #$0D     CARRIAGE RETURN?
0331   0378 27 81           BEQ LOP2A      IF SO, OUTPUT BLOCK AND RE-INIT.  INDEX
0332                *
0333   037A 39              RTS
0334                *
0335   037B CE 03D4 R LWRIT2 LDX #ERR5      BUFFER OVER-RUN
0336                        PRTMSG
0337 +  037E 3F              SWI
0338 +  037F 31              FCB 49
0339   0380 C6 12           LDA B #18
0340   0382 CE 012C R       LDX #OUTFCB    RETURN ERROR STATUS
0341   0385 E7 05           STA B FCBSTA,X
0342   0387 39              RTS
0343                *
0344                *
0345                *
0346   0388 BD 02DE R SOPEN  JSR LOPEN      SPECIAL OPEN FOR LINE-PRINTER
0347   038B 86 0C           LDA A #$0C     ISSUE A LINEFEED
0348   038D 20 CC           BRA LWRITE
0349                *
0350                *
0351                *
0352   038F 39        SCLOSE RTS           SPECIAL CLOSE FOR LINEPRINTER
0353                *
0354                *
0355   0390 20        ERR1   FCC ' BAD INPUT'
0356   039A 0D               FCB $0D
0357                *
0358   039B 20        ERR2   FCC ' BAD OUTPUT'
0359   03A6 0D               FCB $0D
0360                *
0361   03A7 20        ERR3   FCC ' ILLEGAL INPUT DEVICE'
0362   03BC 0D               FCB $0D
0363                *
0364   03BD 20        ERR4   FCC ' ILLEGAL OUTPUT DEVICE'
0365   03D3 0D               FCB $0D
0366                *
0367   03D4 20        ERR5   FCC ' BUFFER OVER-RUN'
0368   03E4 0D               FCB $0D
0369                *
0370   03E5 20        NUMBER FCC ' NUMBER ERROR'
0371   03F2 0D               FCB $0D
0372                *
0373   03F3 20        FORMAT FCC ' FORMAT ERROR'
0374   0400 0D               FCB $0D
0375                *
0376   0401 20        SWITCH FCC ' ILLEGAL SWITCH'
0377   0410 0D               FCB $0D
0379                * LOOK UP DEVICE IN 'DEVTAB'
0380                *
0381                * INDEX REGISTER RETURNS ADDRESS OF DEVICE BLOCK
0382                * CARRY-BIT CLEAR IF FOUND, SET IF NOT FOUND
0383                *
0384                * WORKS ON PRESENT TOKEN IN CLI
0385                *
0386   0411 DE 20   DLKUP  LDX DESCRA           POINT TO CLI NAME
0387                       PSHX
0388 +  0413 3F             SWI
0389 +  0414 05             FCB 5
0390   0415 CE 025A R       LDX #DEVTAB         POINT TO DEVICE TABLE
0391                       PSHX
0392 +  0418 3F             SWI
0393 +  0419 05             FCB 5
0394   041A FF 0455 R       STX SAVEX
0395   041D D6 22           LDA B DESCRC        GET LENGTH OF NAME
0396   041F C1 03           CMP B #3            DEVICE NAMES ARE 3 CHARS
0397   0421 26 23           BNE NOTFND
0398                *
0399                DLKUP2 CMPC                COMPARE NAMES
0400 +  0423 3F             SWI
0401 +  0424 12             FCB 18
0402   0425 27 25           BEQ FOUND
0403                *
0404   0427 FE 0455 R       LDX SAVEX           GET NEXT TABLE ENTRY
0405   042A C6 0B           LDA B #11           11 BYTES/ ENTRY
0406                       ADDBX
0407 +  042C 3F             SWI
0408 +  042D 0A             FCB 10
0409   042E FF 0455 R       STX SAVEX
0410   0431 A6 00           LDA A 0,X           CHECK FOR END OF TABLE
0411   0433 27 11           BEQ NOTFND
0412                *
0413   0435 31              INS
0414   0436 31              INS                 CLEAN STACK
0415   0437 31              INS
0416   0438 31              INS
0417   0439 DE 20           LDX DESCRA
0418                       PSHX                RE-SET FOR COMPARE
0419 +  043B 3F             SWI
0420 +  043C 05             FCB 5
0421   043D FE 0455 R       LDX SAVEX
0422                       PSHX
0423 +  0440 3F             SWI
0424 +  0441 05             FCB 5
0425   0442 C6 03           LDA B #3
0426   0444 20 DD           BRA DLKUP2
0427                *
0428   0446 31      NOTFND INS
```

```
0429   0447 31              INS                 CLEAN STACK
0430   0448 31              INS
0431   0449 31              INS
0432   044A 0D              SEC
0433   044B 39              RTS
0434                *
0435   044C 31      FOUND   INS                 CLEAN STACK
0436   044D 31              INS
0437   044E 31              INS
0438   044F 31              INS
0439   0450 FE 0455 R       LDX SAVEX
0440   0453 0C              CLC
0441   0454 39              RTS
0442                *
0443   0455 0002    SAVEX   RMB 2               TEMP. STORAGE
0444   0457 0001    SAVEA   RMB 1
0445                * COMMAND PARSING ROUTINES
0446                *
0447   0458 CE 027B R PIP   LDX #DSKDEV
0448   045B FF 002A R       STX INHND           DEFAULT INPUT DEVICE=DSK
0449   045E FF 0156 R       STX OUTHND          DEFAULT OUTPUT DEVICE=DSK
0450   0461 CE 0000 R       LDX #INFCB
0451   0464 6F 29           CLR FCBSCF,X        NO COMPRESSION ON INPUT
0452   0466 CE 012C R       LDX #OUTFCB
0453   0469 6F 09           CLR FCBDRV,X        DEFAULT DRIVE=0
0454   046B 6F 29           CLR FCBSCF,X        NO COMPRESSION ON OUTPUT
0455   046D 7F 0798 R       CLR PIPFLG          INIT. PARSE FLAG
0456   0470 7F 0259 R       CLR BFLAG           NO BINARY REFORMAT
0457   0473 7F 0258 R       CLR HFLAG           NO HEX REFORMAT
0458   0476 6F 05           CLR FCBSTA,X        NO ERRORS
0459   0478 86 44           LDA A #'D
0460   047A A7 02           STA A FCBGDT,X
0461   047C 86 53           LDA A #'S
0462   047E A7 03           STA A FCBGDT+1,X
0463   0480 86 4B           LDA A #'K
0464   0482 A7 04           STA A FCBGDT+2,X
0465   0484 86 20           LDA A #$20
0466   0486 A7 10           STA A FCBNAM,X      NO FILE NAME (BLANK FIRST)
0467   0488 86 00           LDA A #0            DEFAULT FILETYPE=BINARY (00)
0468   048A A7 1D           STA A FCBTYP,X
0469                        NXTOK               GET A TOKEN FROM CLI
0470 +  048C 3F              SWI
0471 +  048D 2F              FCB 47
0472   048E DE 20           LDX DESCRA
0473   0490 A6 00           LDA A 0,X           CHECK FOR ESCAPE
0474   0492 91 43           CMP A ES
0475   0494 26 01           BNE PIP1
0476                *
0477   0496 39              RTS                 ESCAPE--DONE
0478                *
0479   0497 D6 25   PIP1    LDA B RC            CHECK RC
0480   0499 C1 03           CMP B #3            NUMBER?
0481   049B 26 36           BNE PIP2            NO
0482                *
0483   049D 7D 0027         TST VALUE           VALID DRIVE NO.?
0484   04A0 26 0D           BNE PIP1A           NO
0485                *
0486   04A2 96 28           LDA A VALUE+1       VALID DRIVE NO.?
0487   04A4 81 03           CMP A #3
0488   04A6 22 07           BHI PIP1A           NO
0489                *
0490   04A8 CE 012C R       LDX #OUTFCB
0491   04AB A7 09           STA A FCBDRV,X      STORE DRIVE NO.
0492   04AD 20 08           BRA PIP1B
0493                *
0494   04AF CE 03E5 R PIP1A LDX #NUMBER         NUMBER ERROR
0495                        PRTMSG
0496 +  04B2 3F              SWI
0497 +  04B3 31              FCB 49
0498   04B4 7E 06F3 R       JMP PIPNXT          GET NEW CLI
0499                *
0500                PIP1B   NXTOK               GET A TOKEN
0501 +  04B7 3F              SWI
0502 +  04B8 2F              FCB 47
0503   04B9 D6 25           LDA B RC            CHECK RC
0504   04BB C1 3A           CMP B #':           COLON?
0505   04BD 26 F0           BNE PIP1A           NO, ERROR
0506                *
0507                        NXTOK               GET A TOKEN
0508 +  04BF 3F              SWI
0509 +  04C0 2F              FCB 47
0510   04C1 D6 25           LDA B RC            CHECK RC
0511   04C3 C1 3D           CMP B #'=           EQUALS?
0512   04C5 26 06           BNE PIP1C           NO
0513                *
0514   04C7 73 0798 R       COM PIPFLG          SET "COPY" MODE FLAG
0515   04CA 7E 0597 R       JMP PIP5            PROCESS INPUT
0516                *
0517   04CD C1 01   PIP1C   CMP B #1            NAME?
0518   04CF 26 06           BNE PIP2A           NO, ERROR
0519                *
0520   04D1 20 28           BRA PIP3            YES, (AND NOT DEVICE)
0521                *
0522   04D3 C1 01   PIP2    CMP B #1            NAME?
0523   04D5 27 08           BEQ PIP2B           YES
0524                *
0525   04D7 CE 03F3 R PIP2A LDX #FORMAT         NO, FORMAT ERROR
0526                        PRTMSG
0527 +  04DA 3F              SWI
0528 +  04DB 31              FCB 49
0529   04DC 7E 06F3 R       JMP PIPNXT          GET NEW CLI
0530                *
0531   04DF BD 0411 R PIP2B JSR DLKUP           DEVICE NAME?
0532   04E2 25 17           BCS PIP3            NO, TRY AS FILE
0533                *
0534   04E4 FF 0156 R       STX OUTHND          SAVE ADDRESS
0535   04E7 CE 012E R       LDX #OUTFCB+FCBGDT
0536                        PSHX
0537 +  04EA 3F              SWI
0538 +  04EB 05              FCB 5
0539   04EC FE 0156 R       LDX OUTHND
0540                        PSHX
0541 +  04EF 3F              SWI
0542 +  04F0 05              FCB 5
0543   04F1 C6 03           LDA B #3            PUT DEVICE NAME INTO FCB
0544                        MOVC
0545 +  04F3 3F              SWI
0546 +  04F4 11              FCB 17
0547   04F5 31              INS
0548   04F6 31              INS                 CLEAN STACK
0549   04F7 31              INS
0550   04F8 31              INS
0551
```

```
0552   04F9 20 35          BRA PIP4              NOW OPEN DEVICE
0553                *
0554   04FB DE 20     PIP3 LDX DESCRA            SAVE POINTER TO NAME
0555   04FD FF 0455 R       STX SAVEX
0556   0500 96 22           LDA A DESCRC          SAVE LENGTH
0557   0502 B7 0457 R       STA A SAVEA
0558                        NXTOK                 GET A TOKEN
0559 +  0505 3F              SWI
0560 +  0506 2F              FCB 47
0561   0507 D6 25           LDA B RC              CHECK RC
0562   0509 C1 2E           CMP B #'.             PERIOD?
0563   050B 26 CA           BNE PIP2A             NO, ERROR
0564                *
0565   050D 7C 0457 R       INC SAVEA             COUNT PERIOD
0566                        NXTOK                 GET A TOKEN
0567 +  0510 3F              SWI
0568 +  0511 2F              FCB 47
0569   0512 D6 25           LDA B RC              CHECK RC
0570   0514 C1 01           CMP B #1              NAME?
0571   0516 26 BF           BNE PIP2A             NO, ERROR
0572                *
0573   0518 D6 22           LDA B DESCRC          GET LENGTH OF EXT
0574   051A FB 0457 R       ADD B SAVEA           TOTAL LENGTH
0575   051D CE 013C R       LDX #OUTFCB+FCBNAM
0576                        PSHX                  POINT TO FCB NAME
0577 +  0520 3F              SWI
0578 +  0521 05              FCB 5
0579   0522 FE 0455 R       LDX SAVEX             POINT TO CLI NAME
0580                        PSHX
0581 +  0525 3F              SWI
0582 +  0526 05              FCB 5
0583                        FMTS                  FORMAT NAME INTO FCB
0584 +  0527 3F              SWI
0585 +  0528 34              FCB 52
0586   0529 31              INS
0587   052A 31              INS                   CLEAN STACK
0588   052B 31              INS
0589   052C 31              INS
0590   052D 5D              TST B                 ERRORS?
0591   052E 26 A7           BNE PIP2A             YES
0592                *
0593                PIP4    NXTOK                 GET A TOKEN
0594 +  0530 3F              SWI
0595 +  0531 2F              FCB 47
0596   0532 D6 25           LDA B RC              CHECK RC
0597   0534 C1 2F           CMP B #'/             SWITCH INDICATOR?
0598   0536 26 3C           BNE PIP4E             NO
0599                *
0600                        NXTOK                 GET SWITCH FROM CLI
0601 +  0538 3F              SWI
0602 +  0539 2F              FCB 47
0603   053A DE 20           LDX DESCRA
0604   053C A6 00           LDA A 0,X
0605   053E CE 012C R       LDX #OUTFCB
0606   0541 81 42           CMP A #'B             "B"?
0607   0543 26 07           BNE PIP4A             NO
0608                *
0609   0545 7C 0259 R       INC BFLAG             SET BINARY FLAG
0610   0548 6F 1D           CLR FCBTYP,X          FILETYPE=BINARY (00)
0611   054A 20 E4           BRA PIP4
0612                *
0613   054C 81 43     PIP4A CMP A #'C             "C"?
0614   054E 26 08           BNE PIP4B             NO
0615                *
0616   0550 86 03           LDA A #$03            FILETYPE=TEXT (03)
0617   0552 A7 1D           STA A FCBTYP,X
0618   0554 6C 29           INC FCBSCF,X          SET SPACE-COMPRESSION ON
0619   0556 20 D8           BRA PIP4
0620                *
0621   0558 81 48     PIP4B CMP A #'H             "H"?
0622   055A 26 09           BNE PIP4C             NO
0623                *
0624   055C 7C 0258 R       INC HFLAG             SET HEX FLAG
0625   055F 86 03           LDA A #$03            FILETYPE=TEXT (03)
0626   0561 A7 1D           STA A FCBTYP,X
0627   0563 20 CB           BRA PIP4
0628                *
0629   0565 81 54     PIP4C CMP A #'T             "T"?
0630   0567 26 06           BNE PIP4D             NO
0631                *
0632   0569 86 03           LDA A #$03            FILETYPE=TEXT (03)
0633   056B A7 1D           STA A FCBTYP,X
0634   056D 20 C1           BRA PIP4
0635                *
0636   056F CE 0401 R PIP4D LDX #SWITCH           ILLEGAL SWITCH ERROR
0637   0572 20 64           BRA PIP5B
0638                *
0639   0574 CE 012C R PIP4E LDX #OUTFCB
0640                        TXAB                  PASS FCB ADDRESS
0641 +  0577 3F              SWI
0642 +  0578 02              FCB 2
0643   0579 FE 0156 R       LDX OUTHND
0644   057C EE 03           LDX 3,X               GET ADDRESS OF HANDLER
0645   057E AD 00           JSR 0,X               OPEN DEVICE OR FILE
0646   0580 CE 012C R       LDX #OUTFCB
0647   0583 6D 05           TST FCBSTA,X          CHECK STATUS
0648   0585 27 08           BEQ PIP4F             GOOD
0649                *
0650   0587 CE 039B R       LDX #ERR2             BAD OUTPUT ERROR
0651                        PRTMSG
0652 +  058A 3F              SWI
0653 +  058B 31              FCB 49
0654   058C 7E 06F3 R       JMP PIPNXT            GET NEW CLI
0655                *
0656   058F DE 20     PIP4F LDX DESCRA            RECOVER TOKEN
0657   0591 E6 00           LDA B 0,X
0658   0593 C1 3D           CMP B #'=             EQUALS?
0659   0595 26 73           BNE PIP6A             NO, ERROR
0660                *
0661                * NOW PROCESS INPUT SIDE OF CLI
0662                *
0663   0597 CE 027B R PIP5  LDX #DSKDEV
0664   059A FF 002A R       STX INHND             DEFAULT DEVICE=DSK
0665   059D CE 0000 R       LDX #INFCB
0666   05A0 6F 09           CLR FCBDRV,X          DEFAULT DRIVE=0
0667   05A2 6F 05           CLR FCBSTA,X          NO ERRORS
0668   05A4 86 44           LDA A #'D
0669   05A6 A7 02           STA A FCBGDT,X
0670   05A8 86 53           LDA A #'S
0671   05AA A7 03           STA A FCBGDT+1,X
0672   05AC 86 4B           LDA A #'K
0673   05AE A7 04           STA A FCBGDT+2,X
```

```
0674   05B0 86 20          LDA A #$20
0675   05B2 A7 10          STA A FCBNAM,X      NO FILE NAME
0676                       NXTOK               GET A TOKEN
0677 +  05B4 3F             SWI
0678 +  05B5 2F             FCB 47
0679   05B6 D6 25          LDA B RC            CHECK RC
0680   05B8 C1 03          CMP B #3            NUMBER?
0681   05BA 27 07          BEQ PIP5D           YES
0682                *
0683   05BC 7D 0798 R       TST PIPFLG          IN "COPY" MODE?
0684   05BF 27 45          BEQ PIP6            NO, O.K.
0685                *
0686   05C1 20 47          BRA PIP6A           IN "COPY" MODE, MUST HAVE NUMBER
0687                *
0688   05C3 7D 0027   PIP5D TST VALUE           VALID DRIVE NO.?
0689   05C6 26 0D          BNE PIP5A           NO
0690                *
0691   05C8 96 28          LDA A VALUE+1       VALID DRIVE NO.?
0692   05CA 81 03          CMP A #3
0693   05CC 22 07          BHI PIP5A           NO
0694                *
0695   05CE CE 0000 R       LDX #INFCB
0696   05D1 A7 09          STA A FCBDRV,X      SAVE DRIVE NO.
0697   05D3 20 14          BRA PIP5D
0698                *
0699   05D5 CE 03E5 R PIP5A LDX #NUMBER         NUMBER ERROR
0700                  PIP5B PRTMSG
0701 +  05D8 3F             SWI
0702 +  05D9 31             FCB 49
0703   05DA CE 012C R       LDX #OUTFCB
0704                       TXAB                PASS FCB ADDRESS
0705 +  05DD 3F             SWI
0706 +  05DE 02             FCB 2
0707   05DF FE 0156 R       LDX OUTHND          GET CLOSE HANDLER
0708   05E2 EE 05          LDX 5,X
0709   05E4 AD 00          JSR 0,X             CLOSE OUTPUT FILE
0710                *
0711   05E6 7E 06F3 R PIP5C JMP PIPNXT          GET NEW CLI
0712                *
0713                  PIP5D NXTOK               GET A TOKEN
0714 +  05E9 3F             SWI
0715 +  05EA 2F             FCB 47
0716   05EB D6 25          LDA B RC            CHECK RC
0717   05ED C1 3A          CMP B #':           COLON?
0718   05EF 26 E4          BNE PIP5A           NO, ERROR
0719                *
0720                       NXTOK               GET A TOKEN
0721 +  05F1 3F             SWI
0722 +  05F2 2F             FCB 47
0723   05F3 D6 25          LDA B RC            CHECK RC
0724   05F5 C1 0D          CMP B #$0D          END-OF-LINE?
0725   05F7 26 03          BNE PIP5E           NO
0726                *
0727   05F9 7E 0719 R       JMP DTDCPY          YES, DISK-TO-DISK COPY
0728                *
0729   05FC C1 01     PIP5E CMP B #1            UNAMBIG. NAME?
0730   05FE 27 2B          BEQ PIP7            YES
0731                *
0732   0600 C1 02          CMP B #2            WILD-CARD NAME?
0733   0602 26 06          BNE PIP6A           IF NOT, ERROR
0734                *
0735   0604 20 25          BRA PIP7
0736                *
0737   0606 C1 01     PIP6  CMP B #1            NAME?
0738   0608 27 05          BEQ PIP6B           YES
0739                *
0740   060A CE 03F3 R PIP6A LDX #FORMAT         FORMAT ERROR
0741   060D 20 C9          BRA PIP5B           FINISH AND CLOSE OUTPUT FILE
0742                *
0743   060F BD 0411 R PIP6B JSR DLKUP           DEVICE NAME?
0744   0612 25 17          BCS PIP7            NO, TRY AS FILE NAME
0745                *
0746   0614 FF 002A R       STX INHND           SAVE ADDRESS
0747   0617 CE 0002 R       LDX #INFCB+FCBGDT
0748                       PSHX
0749 +  061A 3F             SWI
0750 +  061B 05             FCB 5
0751   061C FE 002A R       LDX INHND
0752                       PSHX
0753 +  061F 3F             SWI
0754 +  0620 05             FCB 5
0755   0621 C6 03          LDA B #3            PUT NAME INTO FCB
0756                       MOVC
0757 +  0623 3F             SWI
0758 +  0624 11             FCB 17
0759   0625 31             INS
0760   0626 31             INS                 CLEAN STACK
0761   0627 31             INS
0762   0628 31             INS
0763   0629 20 46          BRA PIP8            NOW OPEN DEVICE
0764                *
0765   062B DE 20     PIP7  LDX DESCRA
0766   062D FF 0455 R       STX SAVEX           SAVE POINTER TO NAME
0767   0630 96 22          LDA A DESCRC
0768   0632 B7 0457 R       STA A SAVEA         SAVE LENGTH
0769                       NXTOK               GET A TOKEN
0770 +  0635 3F             SWI
0771 +  0636 2F             FCB 47
0772   0637 D6 25          LDA B RC            CHECK RC
0773   0639 C1 2E          CMP B #'.           PERIOD?
0774   063B 26 CD          BNE PIP6A           NO, ERROR
0775                *
0776   063D 7C 0457 R       INC SAVEA           COUNT PERIOD
0777                       NXTOK               GET A TOKEN
0778 +  0640 3F             SWI
0779 +  0641 2F             FCB 47
0780   0642 D6 25          LDA B RC            CHECK RC
0781   0644 C1 01          CMP B #1            UNAMBIG. NAME?
0782   0646 27 04          BEQ PIP7A           YES
0783                *
0784   0648 C1 02          CMP B #2            WILD-CARD NAME?
0785   064A 26 BE          BNE PIP6A           NO, ERROR
0786                *
0787   064C D6 22     PIP7A LDA B DESCRC        GET LENGTH OF EXT
0788   064E FB 0457 R       ADD B SAVEA         TOTAL LENGTH
0789   0651 CE 0010 R       LDX #INFCB+FCBNAM
0790                       PSHX
0791 +  0654 3F             SWI
0792 +  0655 05             FCB 5
0793   0656 FE 0455 R       LDX SAVEX
0794                       PSHX                MOVE NAME INTO FCB
0795 +  0659 3F             SWI
```

```
0796 +  065A 05                FCB 5
0797                           FMTS
0798 +  065B 3F                SWI
0799 +  065C 34                FCB 52
0800    065D 31                INS
0801    065E 31                INS                    CLEAN STACK
0802    065F 31                INS
0803    0660 31                INS
0804    0661 C1 02             CMP B #2               ERROR?
0805    0663 27 A5             BEQ PIP6A              YES
0806                   *
0807    0665 7D 0798 R         TST PIPFLG             IN "COPY" MODE?
0808    0668 27 03             BEQ PIP7B              NO
0809                   *
0810    066A 7E 089A R         JMP FILCPY             PROCESS FILE-COPY
0811                   *
0812    066D C1 01     PIP7B   CMP B #1               WILD-CARD NAME?
0813    066F 27 99             BEQ PIP6A              YES, ERROR
0814                   *
0815    0671 CE 0000 R PIP8    LDX #INFCB
0816                           TXAB                   PASS FCB ADDRESS
0817 +  0674 3F                SWI
0818 +  0675 02                FCB 2
0819    0676 FE 002A R         LDX INHND
0820    0679 EE 03             LDX 3,X
0821    067B AD 00             JSR 0,X                OPEN DEVICE OR FILE
0822                   *
0823    067D CE 0000 R PIP8A   LDX #INFCB
0824                           PRTERR                 PRINT ERROR MESSAGES
0825 +  0680 3F                SWI
0826 +  0681 1E                FCB 30
0827    0682 6D 05             TST FCBSTA,X           CHECK STATUS
0828    0684 27 06             BEQ PIP81              GOOD?
0829                   *
0830    0686 CE 0390 R         LDX #ERR1              BAD INPUT MESSAGE
0831    0689 7E 05D8 R         JMP PIP5B              CLOSE FILES
0832                   *
0833    068C A6 1D     PIP81   LDA A FCBTYP,X         GET FILE TYPE OF INPUT
0834    068E 81 03             CMP A #3               TEXT?
0835    0690 27 08             BEQ PIP82              YES
0836                   *
0837                   * BINARY FILE HERE
0838                   *
0839    0692 7D 0258 R         TST HFLAG              WANT HEX OUTPUT?
0840    0695 27 0D             BEQ PIP8B              NO, COPY AS-IS
0841                   *
0842    0697 7E 0B51 R         JMP HEXFRM             YES, REFORMAT
0843                   *
0844                   * TEXT FILE HERE
0845                   *
0846    069A 6C 29     PIP82   INC FCBSCF,X           SET SPACE-COMPRESS. ON
0847    069C 7D 0259 R         TST BFLAG              WANT BINARY OUTPUT?
0848    069F 27 03             BEQ PIP8B              NO, COPY AS-IS
0849                   *
0850    06A1 7E 0D43 R         JMP BINFRM             YES, REFORMAT
0851                   *
0852    06A4 FE 002A R PIP8B   LDX INHND
0853    06A7 EE 07             LDX 7,X                GET READ HANDLER OF DEVICE
0854    06A9 AD 00             JSR 0,X                READ A CHARACTER
0855    06AB CE 0000 R         LDX #INFCB
0856    06AE E6 05             LDA B FCBSTA,X         CHECK STATUS
0857    06B0 C1 08             CMP B #8               EOF?
0858    06B2 27 1C             BEQ PIP9               YES
0859                   *
0860    06B4 5D                TST B                  ERROR?
0861    06B5 27 03             BEQ PIP8C              NO
0862                   *
0863                           PRTERR                 PRINT ERROR MESSAGES
0864 +  06B7 3F                SWI
0865 +  06B8 1E                FCB 30
0866    06B9 4F                CLR A                  SEND NULL CHARACTER
0867                   *
0868    06BA FE 0156 R PIP8C   LDX OUTHND
0869    06BD EE 09             LDX 9,X                GET WRITE HANDLER OF DEVICE
0870    06BF AD 00             JSR 0,X                WRITE A CHARACTER
0871    06C1 CE 012C R         LDX #OUTFCB
0872    06C4 6D 05             TST FCBSTA,X           CHECK STATUS
0873    06C6 27 DC             BEQ PIP8B              IF GOOD, LOOP
0874                   *
0875                           PRTERR                 PRINT ERROR MESSAGE
0876 +  06C8 3F                SWI
0877 +  06C9 1E                FCB 30
0878    06CA CE 039B R         LDX #ERR2              BAD OUTPUT MESSAGE
0879    06CD 7E 05D8 R         JMP PIP5B              CLOSE OUTFILE AND RE-PROMPT
0880                   *
0881                   PIP9    TXAB                   POINT TO INPUT FCB
0882 +  06D0 3F                SWI
0883 +  06D1 02                FCB 2
0884    06D2 FE 002A R         LDX INHND
0885    06D5 EE 05             LDX 5,X                GET INPUT CLOSE ROUTINE
0886    06D7 AD 00             JSR 0,X                CLOSE DEVICE FILE
0887                           NXTOK                  GET A NEW TOKEN
0888 +  06D9 3F                SWI
0889 +  06DA 2F                FCB 47
0890    06DB D6 25             LDA B RC               CHECK RC
0891    06DD C1 0D             CMP B #$0D             C. R. ?
0892    06DF 26 06             BNE PIP9B              NO
0893                   *
0894    06E1 CE 0713 R         LDX #PRMPT2            ISSUE "DONE" PROMPT
0895    06E4 7E 05D8 R         JMP PIP5B              CLOSE OUTFILE AND RE-PROMPT
0896                   *
0897    06E7 96 26     PIP9B   LDA A CLASS            DELIMITER?
0898    06E9 81 04             CMP A #4
0899    06EB 26 03             BNE PIP9C              NO
0900                   *
0901    06ED 7E 0597 R         JMP PIP5               PROCESS NEW INPUT DEVICE
0902                   *
0903    06F0 7E 060A R PIP9C   JMP PIP6A               FORMAT ERROR
0904                   *
0905    06F3 CE 0000 R PIPNXT  LDX #INFCB             POINT TO INPUT FCB
0906                           TXAB
0907 +  06F6 3F                SWI
0908 +  06F7 02                FCB 2
0909    06F8 FE 002A R         LDX INHND
0910    06FB EE 05             LDX 5,X                POINT TO INPUT CLOSE ROUTINE
0911    06FD AD 00             JSR 0,X                CLOSE INPUT DEVICE FILE
0912    06FF CE 070D R PIP10   LDX #PRMPT             ISSUE 'PIP' PROMPT
0913                           PRTMSG
0914 +  0702 3F                SWI
0915 +  0703 31                FCB 49
0916                           GTCMD                  GET NEW CLI LINE
0917 +  0704 3F                SWI
```

```
0918 +  0705 30              FCB 48
0919    0706 DE 20           LDX DESCRA
0920    0708 DF 23           STX CUCHAR          BACK UP ONE TOKEN
0921    070A 7E 0458 R       JMP PIP             LOOP AGAIN
0922                 *
0923    070D 20      PRMPT   FCC ' PIP-'
0924    0712 04              FCB $4
0925                 *
0926    0713 20      PRMPT2  FCC ' DONE'
0927    0718 0D              FCB $0D
0929                 * DISK-TO-DISK (NON-PACKING) COPY
0930                 *
0931                 * SYNTAX:    PIP TODRV:=FRMDRV:
0932                 *
0933    0719 CE 0000 R DTDCPY LDX #INFCB          POINT TO "FROM" FCB
0934    071C A6 09           LDA A FCBDRV,X      GET DRIVE NO.
0935    071E 8B 30           ADD A #$30          MAKE ASCII
0936    0720 B7 07AB R       STA A FRMDRV
0937    0723 CE 012C R       LDX #OUTFCB         POINT TO "TO" FCB
0938    0726 A6 09           LDA A FCBDRV,X      GET DRIVE NO.
0939    0728 8B 30           ADD A #$30          MAKE ASCII
0940    072A B7 07B6 R       STA A TODRV
0941    072D CE 0799 R       LDX #DTDL1          ISSUE "COPY" MESSAGE
0942                         PRTMSG
0943 +  0730 3F              SWI
0944 +  0731 31              FCB 49
0945                         GTCMD               GET USER RESPONSE
0946 +  0732 3F              SWI
0947 +  0733 30              FCB 48
0948    0734 DE 20           LDX DESCRA
0949    0736 A6 00           LDA A 0,X
0950    0738 81 59           CMP A #'Y           "YES"?
0951    073A 26 59           BNE DTDCP4          NO, SKIP OVER
0952                 *
0953    073C 86 00           LDA A #FSTTRK       INIT. (A,B) TO FIRST T/S
0954    073E C6 01           LDA B #FSTSEC
0955                 *
0956    0740 CE 012C R DTDCP1 LDX #OUTFCB        POINT TO "TO" FCB
0957    0743 A7 0A           STA A FCBTRK,X      INIT. T/S
0958    0745 E7 0B           STA B FCBSCT,X
0959    0747 CE 0000 R       LDX #INFCB          POINT TO "FROM" FCB
0960    074A A7 0A           STA A FCBTRK,X      INIT. T/S
0961    074C E7 0B           STA B FCBSCT,X
0962                         IOHDR               READ "FROM" SECTOR
0963 +  074E 3F              SWI
0964 +  074F 13              FCB 19
0965    0750 6D 05           TST FCBSTA,X        CHECK STATUS
0966    0752 27 07           BEQ DTDCP2          O.K.
0967                 *
0968                         PRTERR              PRINT ERROR MESSAGE
0969 +  0754 3F              SWI
0970 +  0755 1E              FCB 30
0971    0756 CE 07CB R       LDX #DRERR          "READ"
0972                         PRTMSG
0973 +  0759 3F              SWI
0974 +  075A 31              FCB 49
0975                 *
0976    075B CE 0158 R DTDCP2 LDX #OUTBUF        POINT TO OUTPUT SECTOR BUFFER
0977                         PSHX
0978 +  075E 3F              SWI
0979 +  075F 05              FCB 5
0980    0760 CE 002C R       LDX #INBUF          POINT TO INPUT SECTOR BUFFER
0981                         PSHX
0982 +  0763 3F              SWI
0983 +  0764 05              FCB 5
0984    0765 C6 80           LDA B #SECSIZ       MOVE DATA FROM INPUT TO OUTPUT
0985                         MOVC
0986 +  0767 3F              SWI
0987 +  0768 11              FCB 17
0988    0769 31              INS
0989    076A 31              INS                 CLEAN STACK
0990    076B 31              INS
0991    076C 31              INS
0992    076D CE 012C R       LDX #OUTFCB         POINT TO "TO" FCB
0993                         IOHDR               WRITE "TO" SECTOR
0994 +  0770 3F              SWI
0995 +  0771 13              FCB 19
0996    0772 6D 05           TST FCBSTA,X        CHECK STATUS
0997    0774 27 07           BEQ DTDCP3          O.K.
0998                 *
0999                         PRTERR              PRINT ERROR MESSAGE
1000 +  0776 3F              SWI
1001 +  0777 1E              FCB 30
1002    0778 CE 07D7 R       LDX #DWERR          "WRITE"
1003                         PRTMSG
1004 +  077B 3F              SWI
1005 +  077C 31              FCB 49
1006                 *
1007    077D CE 0000 R DTDCP3 LDX #INFCB
1008    0780 A6 0A           LDA A FCBTRK,X      RECOVER T/S
1009    0782 E6 0B           LDA B FCBSCT,X
1010    0784 5C              INC B               NEXT SECTOR
1011    0785 C1 1A           CMP B #TRKSIZ       END OF TRACK?
1012    0787 26 B7           BNE DTDCP1          IF NOT, LOOP
1013                 *
1014    0789 C6 01           LDA B #FSTSEC       IF SO, FIRST SECTOR
1015    078B 4C              INC A               NEXT TRACK
1016    078C 81 4D           CMP A #DSKSIZ       END OF DISK?
1017    078E 26 B0           BNE DTDCP1          IF NOT, LOOP
1018                 *
1019    0790 CE 07BB R       LDX #DTDL2          ISSUE "DONE"
1020                         PRTMSG
1021 +  0793 3F              SWI
1022 +  0794 31              FCB 49
1023    0795 7E 06FF R DTDCP4 JMP PIP10          GET NEW CLI LINE
1024                 *
1025    0798 0001    PIPFLG  RMB 1               PARSING FLAG
1026                 *
1027    0799 0A      DTDL1   FCB $0A
1028    079A 20              FCC ' COPY FROM DRIVE '
1029    07AB 0001    FRMDRV  RMB 1
1030    07AC 20              FCC ' TO DRIVE '
1031    07B6 0001    TODRV   RMB 1
1032    07B7 20              FCC ' ? '
1033    07BA 04              FCB $04
1034                 *
1035    07BB 20      DTDL2   FCC ' COPY COMPLETE'
1036    07C9 0A0D            FDB $0A0D
1037                 *
1038    07CB 20      DRERR   FCC ' READ ERROR'
1039    07D6 0D              FCB $0D
1040    07D7 20      DWERR   FCC ' WRITE ERROR'
```

```
07E3 0D                 FCB $0D
                        *
                        * FILE-COPY (PACKING) WITH WILD-CARD CAPABILITY
                        *
                        * SYNTAX:     PIP TODRV:=FRMDRV:FILE.EXT
                        * (WHERE "FILE" AND "EXT" MAY USE WILD-CARDS)
                        *
                        * EXTRA FCB FOR FILE COPY
                        *
07E4 0002       CPYFCB  RMB 2
07E6 44                 FCC 'DSK'
07E9 0002               RMB 2
07EB 080E     R         FDB CPYBUF
07ED 0021               RMB 33
080E 0080       CPYBUF  RMB SECSIZ
                        *
088E 000C       TMPBUF  RMB 12          STORAGE FOR FILENAME (WC)
                        *
                        *
089A CE 088E R  FILCPY  LDX #TMPBUF
                        PSHX
 089D 3F                 SWI
 089E 05                 FCB 5
089F CE 0010 R          LDX #INFCB+FCBNAM
                        PSHX
 08A2 3F                 SWI
 08A3 05                 FCB 5
08A4 C6 0C              LDA B #12
                        MOVC            MOVE WC NAME TO TEMP.
 08A6 3F                 SWI
 08A7 11                 FCB 17
08A8 31                 INS
08A9 31                 INS
08AA 31                 INS             CLEAN STACK
08AB 31                 INS
08AC CE 0000 R          LDX #INFCB
08AF A6 09              LDA A FCBDRV,X
08B1 CE 07E4 R          LDX #CPYFCB     INIT. COPY FCB DRIVE
08B4 A7 09              STA A FCBDRV,X
08B6 6F 06              CLR FCBDTT,X    MAKE INPUT
08B8 6F 29              CLR FCBSCF,X    CLEAR 'FILE-FOUND' MARK
                        OPEND           OPEN DIRECTORY ON INPUT DISK
 08BA 3F                 SWI
 08BB 17                 FCB 23
08BC A6 05      FILCP1  LDA A FCBSTA,X  CHECK STATUS
08BE 27 3C              BEQ FILCP3      GOOD?
                        *
08C0 81 01              CMP A #1        END-OF-DIRECTORY?
08C2 26 1F              BNE FILCP2      NO, ERROR
                        *
08C4 6D 29              TST FCBSCF,X    FOUND A FILE?
08C6 27 03              BEQ *+5         NO, ERROR
                        *
08C8 7E 06D0 R          JMP PIP9        YES, CONTINUE PARSE OF CLI
                        *
08CB CE 08D3 R          LDX #FNFND      FILE-NOT-FOUND ERROR
                        PRTMSG
 08CE 3F                 SWI
 08CF 31                 FCB 49
08D0 7E 06D0 R          JMP PIP9        CONTINUE PARSE OF CLI
                        *
```

```
08D3 20         FNFND   FCC ' FILE NOT FOUND'
08E2 0D                 FCB $0D
                        *
08E3 CE 08EB R  FILCP2  LDX #FERROR     FILE ERROR
                        PRTMSG
 08E6 3F                 SWI
 08E7 31                 FCB 49
08E8 7E 06D0 R          JMP PIP9        CONTINUE PARSE OF CLI
                        *
08EB 20         FERROR  FCC ' DIRECTORY ERROR'
08FB 0D                 FCB $0D
                        *
08FC EE 27      FILCP3  LDX FCBIND,X    POINT TO DIR. NAME FIELD
08FE A6 00              LDA A 0,X
0900 81 20              CMP A #$20      FIRST CHAR.=BLANK?
0902 27 11              BEQ FILCP4      YES, SKIP FILE
                        *
                        PSHX
 0904 3F                 SWI
 0905 05                 FCB 5
0906 CE 088E R          LDX #TMPBUF
                        PSHX
 0909 3F                 SWI
 090A 05                 FCB 5
090B C6 0C              LDA B #12
                        CMWC            -COMPARE DIR. NAME TO CLI NAME (WC)
 090D 3F                 SWI
 090E 35                 FCB 53
090F 31                 INS
0910 31                 INS             CLEAN STACK
0911 31                 INS
0912 31                 INS
0913 27 07              BEQ FILCP5      FOUND FILE?
                        *
0915 CE 07E4 R  FILCP4  LDX #CPYFCB
                        GETDR           GET NEW DIRECTORY ENTRY
 0918 3F                 SWI
 0919 1A                 FCB 26
091A 20 A0              BRA FILCP1
                        *
091C CE 07E4 R  FILCP5  LDX #CPYFCB
091F 6C 29              INC FCBSCF,X    MARK FILE FOUND
0921 CE 0A04 R          LDX #CPRMPT     PRINT ' COPY-'
                        PRTMSG
 0924 3F                 SWI
 0925 31                 FCB 49
0926 CE 07E4 R          LDX #CPYFCB
0929 A6 09              LDA A FCBDRV,X
092B 8B 30              ADD A #$30      MAKE DRIVE NO. ASCII
092D B7 0A0C R          STA A DRIVE
0930 CE 0A0C R          LDX #DRIVE
                        PRTMSG          PRINT 'DRIVE:'
 0933 3F                 SWI
 0934 31                 FCB 49
0935 CE 07E4 R          LDX #CPYFCB
0938 EE 27              LDX FCBIND,X    POINT TO FILE NAME IN DIRECTORY
093A 86 04              LDA A #$04
093C A7 0C              STA A 12,X      PUT IN TERMINATOR
                        PRTMSG          PRINT 'FILE.EXT'
 093E 3F                 SWI
 093F 31                 FCB 49
```

```
0940 CE 0A0F R        LDX #QMRK
                      PRTMSG          PRINT ' ?'
0943 3F                SWI
0944 31                FCB 49
                      GTCMD           GET USER RESPONSE
0945 3F                SWI
0946 30                FCB 48
0947 DE 20            LDX DESCRA
0949 A6 00            LDA A 0,X
094B 81 59            CMP A #'Y       "YES"?
094D 26 C6            BNE FILCP4      IF NOT, GET NEW ENTRY
                *
094F CE 0010 R        LDX #INFCB+FCBNAM
                      PSHX
0952 3F                SWI
0953 05                FCB 5
0954 CE 07E4 R        LDX #CPYFCB
0957 EE 27            LDX FCBIND,X
0959 FF 0455 R        STX SAVEX       SAVE DIRECTORY POINTER
                      PSHX
095C 3F                SWI
095D 05                FCB 5
095E C6 0C            LDA B #12
                      MOVC            MOVE NAME TO INPUT FCB
0960 3F                SWI
0961 11                FCB 17
0962 31               INS
0963 31               INS             CLEAN STACK
0964 31               INS
0965 31               INS
0966 CE 013C R        LDX #OUTFCB+FCBNAM
                      PSHX
0969 3F                SWI
096A 05                FCB 5
096B FE 0455 R        LDX SAVEX
                      PSHX
096E 3F                SWI
096F 05                FCB 5
0970 C6 0C            LDA B #12
                      MOVC            MOVE NAME TO OUTPUT FCB
0972 3F                SWI
0973 11                FCB 17
0974 31               INS
0975 31               INS
0976 31               INS             CLEAN STACK
0977 31               INS
0978 CE 0000 R        LDX #INFCB
                      OPEN            OPEN INPUT FILE
097B 3F                SWI
097C 14                FCB 20
097D 6D 05            TST FCBSTA,X    CHECK STATUS
097F 27 1A            BEQ FILCP6      GOOD
                *
0981 CE 0390 R        LDX #ERR1       INPUT ERROR
                      PRTMSG
0984 3F                SWI
0985 31                FCB 49
0986 CE 0000 R        LDX #INFCB
                      CLOSE           FORCE FILE CLOSED
0989 3F                SWI
098A 15                FCB 21
```

```
098B CE 07E4 R        LDX #CPYFCB
098E B6 0455 R        LDA A SAVEX
0991 F6 0456 R        LDA B SAVEX+1
0994 A7 27            STA A FCBIND,X
0996 E7 28            STA B FCBIND+1,X
0998 7E 0915 R        JMP FILCP4      GET NEXT FILE
                *
099B A6 1D      FILCP6 LDA A FCBTYP,X  GET INPUT FILE TYPE
099D CE 012C R        LDX #OUTFCB
09A0 A7 1D            STA A FCBTYP,X  MAKE OUTPUT FILE SAME TYPE
                      OPEN            OPEN OUTPUT FILE
09A2 3F                SWI
09A3 14                FCB 20
09A4 6D 05            TST FCBSTA,X    CHECK STATUS
09A6 27 1C            BEQ FILCP7      GOOD
                *
                      PRTERR          PRINT ERROR MESSAGE
09A8 3F                SWI
09A9 1E                FCB 30
09AA CE 039B R        LDX #ERR2       OUTPUT ERROR
                      PRTMSG
09AD 3F                SWI
09AE 31                FCB 49
09AF CE 012C R        LDX #OUTFCB
                      CLOSE           FORCE FILE CLOSED
09B2 3F                SWI
09B3 15                FCB 21
09B4 CE 07E4 R        LDX #CPYFCB
09B7 B6 0455 R        LDA A SAVEX
09BA F6 0456 R        LDA B SAVEX+1
09BD A7 27            STA A FCBIND,X
09BF E7 28            STA B FCBIND+1,X
09C1 7E 0915 R        JMP FILCP4      GET NEXT FILE
                *
09C4 CE 0000 R FILCP7 LDX #INFCB
                      READ            READ CHAR. FROM INPUT
09C7 3F                SWI
09C8 18                FCB 24
09C9 E6 05            LDA B FCBSTA,X
09CB C1 08            CMP B #8        EOF?
09CD 27 1B            BEQ FILCP9      YES
                *
09CF 5D               TST B           ERROR?
09D0 27 07            BEQ FILCP8      NO
                *
                      PRTERR          PRINT ERROR MESSAGE
09D2 3F                SWI
09D3 1E                FCB 30
09D4 CE 0390 R        LDX #ERR1       BAD INPUT
                      PRTMSG
09D7 3F                SWI
09D8 31                FCB 49
                *
09D9 CE 012C R FILCP8 LDX #OUTFCB
                      WRITE           WRITE CHAR. TO OUTPUT
09DC 3F                SWI
09DD 19                FCB 25
09DE 6D 05            TST FCBSTA,X    CHECK STATUS
09E0 27 E2            BEQ FILCP7      GOOD, LOOP FOR ANOTHER CHAR.
                *
                      PRTERR          PRINT ERROR MESSAGE
```

```
1286 +  09E2 3F              SWI
1287 +  09E3 1E              FCB 30
1288    09E4 CE 039B R       LDX #ERR2      BAD OUTPUT
1289    09E7 7E 05D8 R       JMP PIP5B      FINISH UP
1290                 *
1291    09EA CE 0000 R FILCP9 LDX #INFCB
1292                 CLOSE          CLOSE INPUT
1293 +  09ED 3F              SWI
1294 +  09EE 15              FCB 21
1295    09EF CE 012C R       LDX #OUTFCB
1296                 CLOSE          CLOSE OUTPUT
1297 +  09F2 3F              SWI
1298 +  09F3 15              FCB 21
1299    09F4 CE 07E4 R       LDX #CPYFCB
1300    09F7 B6 0455 R       LDA A SAVEX
1301    09FA F6 0456 R       LDA B SAVEX+1
1302    09FD A7 27           STA A FCBIND,X
1303    09FF E7 28           STA B FCBIND+1,X
1304    0A01 7E 0915 R       JMP FILCP4     GET NEXT FILE
1305                 *
1306                 *
1307    0A04 20      CPRMPT  FCC ' COPY- '
1308    0A08 04              FCB $4
1309                 *
1310    0A0C 0001    DRIVE   RMB 1
1311    0A0D 3A              FCC ':'
1312    0A0E 04              FCB $4
1313                 *
1314    0A0F 20      QMRK    FCC ' ? '
1315    0A12 04              FCB $4
1317                 * REFORMAT FROM BINARY TO HEX (MIKBUG) FORMAT
1318                 *
1319    0A13 0001    FCNT    RMB 1          BYTES IN FRAME
1320    0A14 0001    CHKSUM  RMB 1          CHECKSUM
1321    0A15 0002    ADDRES  RMB 2          ADDRESS FIELD
1322    0A17 0100    TBUF    RMB 256        TEMP.  BUFFER
1323                 *
1324                 * GET A CHARACTER FROM BINARY FORMAT
1325                 *
1326    0B17 FE 002A R GETBIN LDX INHND
1327    0B1A EE 07           LDX 7,X
1328    0B1C AD 00           JSR 0,X        CALL INPUT HANDLER
1329    0B1E CE 0000 R       LDX #INFCB
1330    0B21 E6 05           LDA B FCBSTA,X RETURN STATUS
1331    0B23 39              RTS
1332                 *
1333                 * PUT A CHARACTER IN BINARY FORMAT
1334                 *
1335    0B24 FE 0156 R PUTBIN LDX OUTHND
1336    0B27 EE 09           LDX 9,X
1337    0B29 AD 00           JSR 0,X        CALL OUTPUT HANDLER
1338    0B2B CE 012C R       LDX #OUTFCB
1339    0B2E E6 05           LDA B FCBSTA,X  RETURN STATUS
1340    0B30 39              RTS
1341                 *
1342                 * PUT A BYTE IN HEXADECIMAL FORMAT
1343                 *
1344    0B31 36      PUTHEX  PSH A          SAVE 'A'
1345    0B32 16              TAB
1346    0B33 FB 0A14 R       ADD B CHKSUM   ADD CHAR.  TO CHECKSUM
1347    0B36 F7 0A14 R       STA B CHKSUM
1348    0B39 8D 07           BSR OUTHL      CONVERT LEFT NIBBLE TO ASCII
1349    0B3B 8D E7           BSR PUTBIN     OUTPUT CHARACTER
1350    0B3D 32              PUL A          RECOVER BYTE
1351    0B3E 8D 06           BSR OUTHR      CONVERT RIGHT NIBBLE TO ASCII
1352    0B40 20 E2           BRA PUTBIN     OUTPUT CHARACTER
1353                 *
1354    0B42 44      OUTHL   LSR A          CONVERT LEFT NIBBLE
1355    0B43 44              LSR A
1356    0B44 44              LSR A
1357    0B45 44              LSR A
1358    0B46 84 0F   OUTHR   AND A #$0F     CONVERT RIGHT NIBBLE
1359    0B48 8B 30           ADD A #$30
1360    0B4A 81 39           CMP A #$39     0-9?
1361    0B4C 23 02           BLS *+4        YES
1362                 *
1363    0B4E 8B 07           ADD A #$07     A-F
1364    0B50 39              RTS
1365                 *
1366                 *
1367                 *
1368    0B51 8D C4   HEXFRM  BSR GETBIN     GET BYTE FROM FILE
1369    0B53 5D              TST B          CHECK STATUS
1370    0B54 27 03           BEQ *+5
1371                 *
1372    0B56 7E 0CA4 R HEX1  JMP STATCK     STATUS NONZERO
1373                 *
1374    0B59 81 16           CMP A #$16     XFER-ADDRESS MARK?
1375    0B5B 26 3E           BNE HEX2       NO
1376                 *
1377                 * PROCESS TRANSFER-ADDRESS RECORD
1378                 *
1379    0B5D 86 53           LDA A #'S      OUTPUT 'S0'
1380    0B5F 8D C3           BSR PUTBIN
1381    0B61 5D              TST B
1382    0B62 26 F2           BNE HEX1
1383                 *
1384    0B64 86 30           LDA A #'0
1385    0B66 8D BC           BSR PUTBIN
1386    0B68 5D              TST B
1387    0B69 26 EB           BNE HEX1
1388                 *
1389    0B6B 7F 0A14 R       CLR CHKSUM     INIT.  CHECKSUM ON RECORD
1390    0B6E 86 03           LDA A #3       BYTE-COUNT=3
1391    0B70 8D BF           BSR PUTHEX
1392    0B72 5D              TST B
1393    0B73 26 E1           BNE HEX1
1394                 *
1395    0B75 8D A0           BSR GETBIN     GET ADDRESS-HIGH
1396    0B77 5D              TST B
1397    0B78 26 DC           BNE HEX1
1398                 *
1399    0B7A 8D B5           BSR PUTHEX     OUTPUT ADDRESS-HIGH
1400    0B7C 5D              TST B
1401    0B7D 26 D7           BNE HEX1
1402                 *
1403    0B7F 8D 96           BSR GETBIN     GET ADDRESS-LOW
1404    0B81 5D              TST B
1405    0B82 26 D2           BNE HEX1
1406                 *
1407    0B84 8D AB           BSR PUTHEX     OUTPUT ADDRESS-LOW
1408    0B86 5D              TST B
```

```
0B87 26 CD            BNE HEX1
                      *
0B89 B6 0A14 R HEX1A  LDA A CHKSUM
0B8C 43               COM A           OUTPUT INVERTED CHECKSUM
0B8D 8D A2            BSR PUTHEX
0B8F 5D               TST B
0B90 26 C4            BNE HEX1
                      *
0B92 86 0D            LDA A #$0D      OUTPUT CARRIAGE-RETURN
0B94 8D 8E            BSR PUTBIN
0B96 5D               TST B
0B97 26 BD            BNE HEX1
                      *
0B99 20 B6            BRA HEXFRM      DONE WITH RECORD
                      *
                      *
                      *
0B9B 81 02      HEX2  CMP A #$02      DATA-FRAME HEADER?
0B9D 27 03            BEQ *+5         YES
                      *
0B9F 7E 0CBC R        JMP EOFST       IF NOT, END-FILE
                      *
                      * HANDLE DATA-FRAME HERE
                      *
0BA2 BD 0B17 R        JSR GETBIN      GET ADDRESS-HIGH
0BA5 5D               TST B
0BA6 27 03            BEQ *+5
                      *
0BA8 7E 0CA4 R HEX21  JMP STATCK      NONZERO STATUS
                      *
0BAB B7 0A15 R        STA A ADDRES
0BAE BD 0B17 R        JSR GETBIN      GET ADDRESS-LOW
0BB1 5D               TST B
0BB2 26 F4            BNE HEX21
                      *
0BB4 B7 0A16 R        STA A ADDRES+1
0BB7 BD 0B17 R        JSR GETBIN      GET FRAME COUNT
0BBA 5D               TST B
0BBB 26 EB            BNE HEX21
                      *
0BBD B7 0A13 R        STA A FCNT
0BC0 B7 0457 R        STA A SAVEA
0BC3 CE 0A17 R        LDX #TBUF       INIT. BUFFER POINTER
0BC6 FF 0455 R        STX SAVEX
0BC9 BD 0B17 R HEX2A  JSR GETBIN      GET DATA BYTE
0BCC 5D               TST B
0BCD 26 D9            BNE HEX21
                      *
0BCF FE 0455 R        LDX SAVEX
0BD2 A7 00            STA A 0,X       STORE BYTE IN TEMP.
0BD4 08               INX
0BD5 FF 0455 R        STX SAVEX
0BD8 7A 0457 R        DEC SAVEA       COUNT DOWN
0BDB 26 EC            BNE HEX2A
                      *
0BDD CE 0A17 R        LDX #TBUF       INIT. BUFFER POINTER
0BE0 FF 0455 R        STX SAVEX
0BE3 B6 0A13 R HEX2B  LDA A FCNT
0BE6 81 1E            CMP A #30       DATA FOR FULL FRAME (30 BYTES)?
0BE8 22 50            BHI HEX2D       YES
                      *
                      * OUTPUT PARTIAL FRAME HERE (<30 BYTES)
                      *
0BEA 7F 0A14 R        CLR CHKSUM      INIT. CHECKSUM
0BED 86 53            LDA A #'S       OUTPUT 'S1'
0BEF BD 0B24 R        JSR PUTBIN
0BF2 5D               TST B
0BF3 26 B3            BNE HEX21
                      *
0BF5 86 31            LDA A #'1
0BF7 BD 0B24 R        JSR PUTBIN
0BFA 5D               TST B
0BFB 26 AB            BNE HEX21
                      *
0BFD B6 0A13 R        LDA A FCNT      OUTPUT FRAME COUNT
0C00 8B 03            ADD A #3        ADD OVERHEAD (ADDRESS+CHECKSUM)
0C02 BD 0B31 R        JSR PUTHEX
0C05 5D               TST B
0C06 26 A0            BNE HEX21
                      *
0C08 B6 0A15 R        LDA A ADDRES    OUTPUT ADDRESS
0C0B BD 0B31 R        JSR PUTHEX
0C0E 5D               TST B
0C0F 27 03            BEQ *+5
                      *
0C11 7E 0CA4 R HEX22  JMP STATCK      NONZERO STATUS
                      *
0C14 B6 0A16 R        LDA A ADDRES+1
0C17 BD 0B31 R        JSR PUTHEX
0C1A 5D               TST B
0C1B 26 F4            BNE HEX22
                      *
0C1D B6 0A13 R        LDA A FCNT
0C20 B7 0457 R        STA A SAVEA
0C23 FE 0455 R HEX2C  LDX SAVEX
0C26 A6 00            LDA A 0,X       GET DATA BYTE FROM TEMP.
0C28 08               INX
0C29 FF 0455 R        STX SAVEX
0C2C BD 0B31 R        JSR PUTHEX      OUTPUT IT
0C2F 5D               TST B
0C30 26 DF            BNE HEX22
                      *
0C32 7A 0457 R        DEC SAVEA       COUNT DOWN
0C35 26 EC            BNE HEX2C
                      *
0C37 7E 0B89 R        JMP HEX1A       FINISH UP (CHECKSUM+C. R. )
                      *
                      *
                      * HANDLE FULL FRAME (30 BYTES) HERE
                      *
0C3A 7F 0A14 R HEX2D  CLR CHKSUM      INIT. CHECKSUM
0C3D 86 53            LDA A #'S       OUTPUT 'S1'
0C3F BD 0B24 R        JSR PUTBIN
0C42 5D               TST B
0C43 26 CC            BNE HEX22
                      *
0C45 86 31            LDA A #'1
0C47 BD 0B24 R        JSR PUTBIN
0C4A 5D               TST B
0C4B 26 C4            BNE HEX22
                      *
0C4D 86 1E            LDA A #30       30 BYTES IN FULL FRAME
```

```
                                    STA A SAVEA
                                    ADD A #3        ADD OVERHEAD
                                    JSR PUTHEX
                                    TST B
                                    BNE HEX22
                            *
                                    LDA A FCNT
                                    SUB A #30       REMOVE 30 BYTES
                                    STA A FCNT
                                    LDA A ADDRES    OUTPUT ADDRESS
                                    JSR PUTHEX
                                    TST B
                                    BNE HEX22
                            *
                                    LDA A ADDRES+1
                                    JSR PUTHEX
                                    TST B
                                    BNE HEX22
                            *
                            HEX2E   LDX SAVEX
                                    LDA A 0,X       GET DATA BYTE
                                    INX
                                    STX SAVEX
                                    JSR PUTHEX      OUTPUT IT
                                    TST B
                                    BNE STATCK
                            *
                                    LDX ADDRES      MOVE ADDRESS
                                    INX
                                    STX ADDRES
                                    DEC SAVEA       COUNT DOWN
                                    BNE HEX2E
                            *
                                    LDA A CHKSUM
                                    COM A           OUTPUT INVERTED CHECKSUM
                                    JSR PUTHEX
                                    TST B
                                    BNE STATCK
                            *
                                    LDA A #$0D      OUTPUT CARRIAGE RETURN
                                    JSR PUTBIN
                                    TST B
                                    BNE STATCK
                            *
                                    JMP HEX2B       CONTINUE UNTIL BUFFER EMPTY
                            *
                            *
                            *
                            * END-FILE AND ERROR HANDLED HERE
                            *
                            STATCK  CMP B #8        END-FILE ON INPUT?
                                    BEQ EOFST       YES
                            *
                                    TST FCBDTT,X    INPUT OR OUTPUT ERROR?
                                    BNE OUTERR      OUTPUT
                            *
                                    PRTERR          PRINT ERROR MESSAGE
+                                    SWI
+                                    FCB 30
                                    LDX #ERR1       INPUT BAD
                                    JMP PIP5B
                            *
                            OUTERR  PRTERR          PRINT ERROR MESSAGE
+                                    SWI
+                                    FCB 30
                                    LDX #ERR2       OUTPUT BAD
                                    JMP PIP5B
                            *
                            EOFST   LDX #INFCB
                                    TXAB
+                                    SWI
+                                    FCB 2
                                    LDX INHND       CLOSE INPUT FILE
                                    LDX 5,X
                                    JSR 0,X
                                    NXTOK           GET NEXT CLI TOKEN
+                                    SWI
+                                    FCB 47
                                    LDA B RC        CHECK RC
                                    CMP B #$0D      C.R.?
                                    BNE EOF2        NO
                            *
                                    LDA A #'S       OUTPUT EOF RECORD ( 'S9' )
                                    JSR PUTBIN
                                    TST B
                                    BNE STATCK
                            *
                                    LDA A #'9
                                    JSR PUTBIN
                                    TST B
                                    BNE STATCK
                            *
                                    LDA A #$0D      OUTPUT CARRIAGE RETURN
                                    JSR PUTBIN
                                    TST B
                                    BNE STATCK
                            *
                                    LDX #PRMPT2     "DONE"
                                    JMP PIP5B
                            *
                            EOF2    JMP PIP9B       FINISH UP
                            *
                            *
                            * REFORMAT FROM HEX (MIKBUG) TO BINARY FORMAT
                            *
                            INHEX   JSR GETBIN      GET A BYTE
                                    TST B           ERROR?
                                    BNE INH2        YES
                            *
                                    SUB A #$30      REMOVE ASCII BIAS
                                    BMI HEXBAD      VALID HEX?
                            *
                                    CMP A #$9       0-9?
                                    BLE INH2        YES
                            *
                                    CMP A #$11      A-F?
                                    BMI HEXBAD      NO
                            *
                                    CMP A #$16      A-F?
                                    BGT HEXBAD      NO
                            *
                                    SUB A #$7       REMOVE ASCII BIAS
```

```
0D09 39          INH2   RTS
                 *
0D0A CE 0D12 R HEXBAD LDX #NOTHEX   INVALID CHARACTER
                       PRTMSG
+  0D0D 3F              SWI
+  0D0E 31              FCB 49
0D0F C6 FF             LDA B #$FF   ERROR STATUS
0D11 39                RTS
                 *
0D12 20          NOTHEX FCC ' BAD HEX CHARACTER'
0D24 0D                FCB $0D
                 *
                 *
0D25 8D CA       GETHEX BSR INHEX    GET A HEX DIGIT
0D27 5D                TST B
0D28 26 18             BNE GET2     ERROR?
                 *
0D2A 48                ASL A        MOVE TO LEFT NIBBLE
0D2B 48                ASL A
0D2C 48                ASL A
0D2D 48                ASL A
0D2E B7 0A17 R         STA A TBUF   SAVE IT
0D31 8D BE             BSR INHEX    GET A HEX DIGIT
0D33 5D                TST B
0D34 26 0C             BNE GET2     ERROR?
                 *
0D36 F6 0A17 R         LDA B TBUF
0D39 1B                ABA          MERGE DIGITS
0D3A 16                TAB
0D3B FB 0A14 R         ADD B CHKSUM ADD INTO CHECKSUM
0D3E F7 0A14 R         STA B CHKSUM
0D41 5F                CLR B
0D42 39          GET2  RTS
                 *
                 *
0D43 BD 0B17 R BINFRM  JSR GETBIN   GET A BYTE
0D46 5D                TST B
0D47 27 03             BEQ *+5      ERROR?
                 *
0D49 7E 0E33 R BIN1    JMP STAT2    YES
                 *
0D4C 81 53             CMP A #'S    HEADER MARK?
0D4E 26 F3             BNE BINFRM   NO, KEEP LOOKING
                 *
0D50 BD 0B17 R         JSR GETBIN   GET A BYTE
0D53 5D                TST B
0D54 26 F3             BNE BIN1
                 *
0D56 81 30             CMP A #'0    TRANSFER ADDRESS ('S0')?
0D58 26 5F             BNE BIN2     NO
                 *
                 * HANDLE TRANSFER ADDRESS HERE
                 *
0D5A 7F 0A14 R         CLR CHKSUM   INIT. CHECKSUM
0D5D 8D C6             BSR GETHEX   GET A HEX BYTE
0D5F 5D                TST B
0D60 26 E7             BNE BIN1
                 * NOTE: FRAME COUNT NOT USED
                 *
0D62 8D C1             BSR GETHEX   GET ADDRESS-HIGH
0D64 5D                TST B
0D65 26 E2             BNE BIN1
                 *
0D67 B7 0A15 R         STA A ADDRES
0D6A 8D B9             BSR GETHEX   GET ADDRESS-LOW
0D6C 5D                TST B
0D6D 26 DA             BNE BIN1
                 *
0D6F B7 0A16 R         STA A ADDRES+1
0D72 B6 0A14 R BIN1A   LDA A CHKSUM
0D75 43                COM A
0D76 B7 0457 R         STA A SAVEA  TEST CHECKSUM
0D79 8D AA             BSR GETHEX   GET CHECKSUM
0D7B 5D                TST B
0D7C 26 CB             BNE BIN1
                 *
0D7E B1 0457 R         CMP A SAVEA  GOOD?
0D81 27 16             BEQ BIN1B    YES
                 *
0D83 CE 0D89 R         LDX #CHKERR  NO, ERROR
0D86 7E 05D8 R         JMP PIP5B
                 *
0D89 20          CHKERR FCC ' CHECKSUM ERROR'
0D98 0D                FCB $0D
                 *
0D99 86 16       BIN1B LDA A #$16   OUTPUT BINARY HEADER
0D9B BD 0B24 R         JSR PUTBIN
0D9E 5D                TST B
0D9F 27 03             BEQ *+5
                 *
0DA1 7E 0E33 R BIN1C   JMP STAT2    BAD STATUS
                 *
0DA4 B6 0A15 R         LDA A ADDRES OUTPUT ADDRESS-HIGH
0DA7 BD 0B24 R         JSR PUTBIN
0DAA 5D                TST B
0DAB 26 F4             BNE BIN1C
                 *
0DAD B6 0A16 R         LDA A ADDRES+1 OUTPUT ADDRESS-LOW
0DB0 BD 0B24 R         JSR PUTBIN
0DB3 5D                TST B
0DB4 26 EB             BNE BIN1C
                 *
0DB6 7E 0D43 R         JMP BINFRM   GET NEW DATA FRAME
                 *
                 *
0DB9 81 31       BIN2  CMP A #'1    DATA HEADER ('S1')?
0DBB 26 6F             BNE BIN3     NO
                 *
                 * HANDLE DATA-RECORD HERE
                 *
0DBD 7F 0A14 R         CLR CHKSUM   INIT. CHECKSUM
0DC0 BD 0D25 R         JSR GETHEX   GET FRAME COUNT
0DC3 5D                TST B
0DC4 26 DB             BNE BIN1C
                 *
0DC6 80 03             SUB A #3     REMOVE OVERHEAD BYTES
0DC8 B7 0A13 R         STA A FCNT
0DCB BD 0D25 R         JSR GETHEX   GET ADDRESS-HIGH
0DCE 5D                TST B
0DCF 26 D0             BNE BIN1C
                 *
0DD1 B7 0A15 R         STA A ADDRES
```

```
0DD4 BD 0D25 R        JSR GETHEX    GET ADDRESS-LOW
0DD7 5D               TST B
0DD8 26 C7            BNE BIN1C
                      *
0DDA B7 0A16 R        STA A ADDRES+1
0DDD 86 02            LDA A #$02    OUTPUT DATA-HEADER MARK
0DDF BD 0B24 R        JSR PUTBIN
0DE2 5D               TST B
0DE3 26 BC            BNE BIN1C
                      *
0DE5 B6 0A15 R        LDA A ADDRES  OUTPUT ADDRESS-HIGH
0DE8 BD 0B24 R        JSR PUTBIN
0DEB 5D               TST B
0DEC 26 45            BNE STAT2
                      *
0DEE B6 0A16 R        LDA A ADDRES+1 OUTPUT ADDRESS-LOW
0DF1 BD 0B24 R        JSR PUTBIN
0DF4 5D               TST B
0DF5 26 3C            BNE STAT2
                      *
0DF7 B6 0A13 R        LDA A FCNT    OUTPUT FRAME COUNT
0DFA BD 0B24 R        JSR PUTBIN
0DFD 5D               TST B
0DFE 26 33            BNE STAT2
                      *
0E00 BD 0D25 R BIN2A  JSR GETHEX    GET A DATA BYTE (HEX FORMAT)
0E03 5D               TST B
0E04 26 2D            BNE STAT2
                      *
0E06 BD 0B24 R        JSR PUTBIN    OUTPUT DATA BYTE (BINARY FORMAT)
0E09 5D               TST B
0E0A 26 27            BNE STAT2
                      *
0E0C 7A 0A13 R        DEC FCNT      COUNT DOWN
0E0F 26 EF            BNE BIN2A
                      *
0E11 B6 0A14 R        LDA A CHKSUM
0E14 43               COM A
0E15 B7 0457 R        STA A SAVEA   TEST AGAINST CHECKSUM
0E18 BD 0D25 R        JSR GETHEX    GET CHECKSUM
0E1B 5D               TST B
0E1C 26 15            BNE STAT2
                      *
0E1E B1 0457 R        CMP A SAVEA   GOOD CHECKSUM?
0E21 27 06            BEQ BIN2B     YES
                      *
0E23 CE 0D89 R        LDX #CHKERR   NO
0E26 7E 05D8 R        JMP PIP5B
                      *
0E29 7E 0D43 R BIN2B  JMP BINFRM    GET NEW FRAME
                      *
                      * HANDLE EOF ('S9') HERE
                      *
0E2C 81 39     BIN3   CMP A #'9     'S9'?
0E2E 26 F9            BNE BIN2B     NO, LOOK FOR IT
                      *
0E30 7E 06D0 R BIN3A  JMP PIP9      YES, CLOSE FILE
                      *
0E33 C1 08     STAT2  CMP B #8      EOF ON INPUT?
0E35 27 F9            BEQ BIN3A     YES, CLOSE FILE
                      *
0E37 7E 0CA4 R        JMP STATCK    HANDLE ERROR STATUS
                      *
                      END
```

```
ADDABX 2219 M   EOF2   0CEE R   GTCMD  24F0 M   PIP1B  04B7 R   SAVEX  0455 R
ADDAX  2232 M   EOFST  0CBC R   HEX1   0B56 R   PIP1C  04CD R   SCLOSE 038F R
ADDBX  224B M   ERR1   0390 R   HEX1A  0B89 R   PIP2   04D3 R   SECSIZ 0080
ADDRES 0A15 R   ERR2   039B R   HEX2   0B9B R   PIP2A  04D7 R   SOPEN  0388 R
ADDXAB 2200 M   ERR3   03A7 R   HEX21  0BA8 R   PIP2B  04DF R   STAT2  0E33 R
BASEQU 2A2A M   ERR4   03BD R   HEX22  0C11 R   PIP3   04FB R   STATCK 0CA4 R
BFLAG  0259 R   ERR5   03D4 R   HEX2A  0BC9 R   PIP4   0530 R   SUBABX 227F M
B1N1   0D49 R   ES     0043     HEX2B  0BE3 R   PIP4A  054C R   SUBAX  2299 M
B1N1A  0D72 R   FCBACS 001E     HEX2C  0C23 R   PIP4B  0558 R   SUBBX  22B3 M
B1N1B  0D99 R   FCBBAK 000E     HEX2D  0C3A R   PIP4C  0565 R   SUBXAB 2265 M
B1N1C  0DA1 R   FCBCHN 0029     HEX2E  0C74 R   PIP4D  056F R   SWITCH 0401 R
B1N2   0DB9 R   FCBDRA 0007     HEXBAD 0D0A R   PIP4E  0574 R   TABX   219C M
B1N2A  0E00 R   FCBDEF 2650 M   HEXFRM 0B51 R   PIP4F  058F R   TB     003F
B1N2B  0E29 R   FCBDRV 0009     HFLAG  0258 R   PIP5   0597 R   TBUF   0A17 R
B1N3   0E2C R   FCBDTT 0006     INBUF  002C R   PIP50  05C3 R   TMPBUF 088E R
B1N3A  0E30 R   FCBEQT 0000     INDEX  24BC M   PIP5A  05D5 R   TODRV  07B6 R
B1NFRM 0D43 R   FCBFTS 001F     INFCB  0000 R   PIP5B  05D8 R   TRKSIZ 001A
BMEM   0033     FCBFWD 000C     INH2   0D09 R   PIP5C  05E6 R   TXAB   2183 M
BS     0039     FCBGDT 0002     INHEX  0CF1 R   PIP5D  05E9 R   VALUE  0027
CHAIN  243A M   FCBIND 0027     INHND  002A R   PIP5E  05FC R   WD     003D
CHKERR 0D89 R   FCBLTS 0021     INITDK 253E M   PIP6   0606 R   WRITE  23D2 M
CHKSUM 0A14 R   FCBNAM 0010     IOERR  02BC R   PIP6A  060A R   XABX   21B5 M
CLASS  0026     FCBNFB 0025     IOHDR  2335 M   PIP6B  060F R
CLOSE  2369 M   FCBNMS 0023     LCLOS2 0328 R   PIP7   062B R
CMEM   0037     FCBSCF 0029     LCLOSE 0321 R   PIP7A  064C R
CMPC   231B M   FCBSCT 000B     LDP    0044     PIP7B  066D R
CMWC   2572 M   FCBSTA 0005     LDPCNT 0045     PIP8   0671 R
CPRMPT 0A04 R   FCBTRK 000A     LOADR  246E M   PIP81  068C R
CPYBUF 080E R   FCBTYP 001D     LOP2   02F8 R   PIP82  069A R
CPYFCB 07E4 R   FCNT   0A13 R   LOP2A  02FB R   PIP8A  067D R
CUCHAR 0023     FERROR 08EB R   LOP2B  02FD R   PIP8B  06A4 R
DCLOSE 02BF R   FIBACS 000E     LOP3   031C R   PIP8C  06BA R
DELETE 2420 M   FIBUEF 2940 M   LOPEN  02DE R   PIP9   06D0 R
DESCRA 0020     FIBFTS 000F     LOPOUT 0308 R   PIP9B  06E7 R
DESCRC 0022     FIBLTS 0011     LREAD  0330 R   PIP9C  06F0 R
DEVTAB 025A R   FIBNAM 0000     LREAD2 0343 R   PIPFLG 0798 R
DIV16  2524 M   FIBNMS 0013     LREAD3 0356 R   PIPNXT 06F3 R
DL     003A     FIBTYP 000D     LWD    0046     PIPPER 0000 RN
DLKUP  0411 R   FILCP1 08BC R   LWRIT2 037B R   PRMPT  070D R
DLKUP2 0423 R   FILCP2 08E3 R   LWRITE 035B R   PRMPT2 0713 R
DOPEN  02B3 R   FILCP3 08FC R   MOVC   2301 M   PRTERR 2454 M
DP     003B     FILCP4 0915 R   MOVS   24A2 M   PRTMSG 250A M
DPCNT  003C     FILCP5 091C R   MUL16  22E7 M   PS     0042
DREAD  02C4 R   FILCP6 099B R   MUL8   22CD M   PSHALL 2151 M
DRERR  07CB R   FILCP7 09C4 R   NL     003E     PSHX   21CE M
DRIVE  0A0C R   FILCP8 09D9 R   NOTFND 0446 R   PULLAL 216A M
DSKDEV 027B R   FILCP9 09EA R   NOTHEX 0D12 R   PULX   21E7 M
DSKSIZ 004D     FILCPY 089A R   NUMBER 03E5 R   PUTBIN 0B24 R
DTDCP1 0740 R   FMTFCB 2488 M   NXTOK  24D6 M   PUTDR  2406 M
DTDCP2 075B R   FMTS   2558 M   OPEN   234F M   PUTHEX 0B31 R
DTDCP3 077D R   FNEND  08D3 R   OPEND  239E M   QMRK   0A0F R
DTDCP4 0795 R   FORMAT 03F3 R   OUTBUF 0158 R   RC     0025
DTDCPY 0719 R   FOUND  044C R   OUTERR 0CB4 R   RCBDBA 0007
DTDL1  0799 R   FRETAB 002B     OUTFCB 012C R   RCBDEF 258C M
DTDL2  07BB R   FRMDRV 07AB R   OUTHL  0B42 R   RCBDTT 0006
DWERR  07D7 R   FSTSEC 0001     OUTHND 0156 R   RCBEQT 0000
DWRITE 02D4 R   FSTTRK 0000     OUTHR  0B46 R   RCBGDT 0002
DX     0040     GET2   0D42 R   PIP    0458 R   RCBSTA 0005
EJ     0041     GETBIN 0B17 R   PIP1   0497 R   READ   23B8 M
EMEM   0035     GETDR  23EC M   PIP10  06FF R   REWIND 2384 M
                GETHEX 0D25 R   PIP1A  04AF R   SAVEA  0457 R
```

```
0001   0000 0000    N        NAM SECURITY
0002                         *
0003                         * TRANSIENT COMMAND 'SECURITY' PROCESSOR
0004                         * SYNTAX:  SECURITY [DRIVE:] FILENAME.EXT,VALUE
0005                         * SET FILE ACCESS CODE TO 'VALUE'
0006                         *
0007                         * BLOCK ADDRESSING DEFINITIONS
0008                         *
                                   FCBDEF
0010 +  0000 0000             FCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0011 +  0000 0002             FCBGDT EQU 2          GENERIC DEVICE TYPE
0012 +  0000 0005             FCBSTA EQU 5          STATUS
0013 +  0000 0006             FCBDTT EQU 6          DATA TRANSFER TYPE
0014 +  0000 0007             FCBDBA EQU 7          DATA BUFFER ADDRESS
0015 +  0000 0009             FCBDRV EQU 9          DRIVE NUMBER
0016 +  0000 000A             FCBTRK EQU 10         TRACK NUMBER
0017 +  0000 000B             FCBSCT EQU 11         SECTOR NUMBER
0018 +  0000 000C             FCBFWD EQU 12         FWD LINK TRACK/SECTOR
0019 +  0000 000E             FCBBAK EQU 14         BACK LINK TRACK/SECTOR
0020 +  0000 0010             FCBNAM EQU 16         FILE NAME (8.3+EOT=13))
0021 +  0000 001D             FCBTYP EQU 29         FILE TYPE
0022 +  0000 001E             FCBACS EQU 30         FILE ACCESS CODE
0023 +  0000 001F             FCBFTS EQU 31         FIRST TRACK/SECTOR
0024 +  0000 0021             FCBLTS EQU 33         LAST TRACK/SECTOR
0025 +  0000 0023             FCBNMS EQU 35         NUMBER OF SECTORS
0026 +  0000 0025             FCBNFB EQU 37         NEXT FCB IN ACTIVE CHAIN
0027 +  0000 0027             FCBIND EQU 39         INDEX INTO DATA BUFFER
0028 +  0000 0029             FCBSCF EQU 41         SPACE COMPRESSION FLAG
0029                                FIBDEF
0030 +  0000 0000             FIBNAM EQU 0          FILE NAME (8.3 + EOT=13)
0031 +  0000 000D             FIBTYP EQU 13         FILE TYPE
0032 +  0000 000E             FIBACS EQU 14         FILE ACCESS CODE
0033 +  0000 000F             FIBFTS EQU 15         FIRST TRACK/SECTOR
0034 +  0000 0011             FIBLTS EQU 17         LAST TRACK/SECTOR
0035 +  0000 0013             FIBNMS EQU 19         NUMBER OF SECTORS
0036                         *
0037                         * BASE-PAGE EQUATES
0038                         *
0039   0000 0020             DESCRA EQU   $20
0040   0000 0022             DESCRC EQU   $22
0041   0000 0023             CUCHAR EQU   $23
0042   0000 0025             RC     EQU   $25
0043   0000 0026             CLASS  EQU   $26
0044   0000 0027             VALUE  EQU   $27
0045                         *
0046                         * DISK ATTRIBUTES
0047                         *
0048   0000 0080             SECSIZ EQU   128          128 BYTES/SECTOR
0049                         *
0050                         * FCB FOR TRANSIENT
0051                         *
0052   0000 0002             SYSFCB RMB 2
0053   0002 44                      FCC 'DSK'
0054   0005 0002                    RMB 2
0055   0007 002A    R               FDB BUFFER
0056   0009 0021                    RMB 33
0057   002A 0080             BUFFER RMB SECSIZ
0058                         *
0059   00AA CE 0000 R SECURE LDX #SYSFCB
0060   00AD 6F 09                   CLR FCBDRV,X        DEFAULT DRIVE=0
```

```
0061   00AF 6F 06            CLR FCBDTT,X          INPUT
0062                         NXTOK                 GET TOKEN FROM CLI
0063 +  00B1 3F               SWI
0064 +  00B2 2F               FCB 47
0065   00B3 D6 25            LDA B RC              CHECK RC
0066   00B5 C1 03            CMP B #3              NUMBER?
0067   00B7 26 2F            BNE SEC2              NO
0068                *
0069   00B9 7D 0027          TST VALUE             VALID DRIVE NO.?
0070   00BC 26 0A            BNE SEC1              NO, ERROR
0071                *
0072   00BE 96 28            LDA A VALUE+1         VALID DRIVE NO.?
0073   00C0 81 03            CMP A #3              (4 DRIVES)
0074   00C2 22 04            BHI SEC1              NO
0075                *
0076   00C4 A7 09            STA A FCBDRV,X        SET DRIVE NO.
0077   00C6 20 14            BRA SEC1A
0078                *
0079   00C8 CE 00CE R SEC1   LDX #NUMBER           NUMBER ERROR
0080                         PRTMSG
0081 +  00CB 3F               SWI
0082 +  00CC 31               FCB 49
0083   00CD 39               RTS
0084                *
0085   00CE 20        NUMBER FCC ' NUMBER ERROR'
0086   00DB 0D               FCB $0D
0087                *
0088                  SEC1A  NXTOK                 GET TOKEN FROM CLI
0089 +  00DC 3F               SWI
0090 +  00DD 2F               FCB 47
0091   00DE D6 25            LDA B RC              CHECK RC
0092   00E0 C1 3A            CMP B #':             COLON?
0093   00E2 26 E4            BNE SEC1              IF NOT, ERROR
0094                *
0095                         NXTOK                 GET TOKEN FROM CLI
0096 +  00E4 3F               SWI
0097 +  00E5 2F               FCB 47
0098   00E6 D6 25            LDA B RC              CHECK RC
0099   00E8 C1 01     SEC2   CMP B #1              UNAMBIG. NAME?
0100   00EA 27 14            BEQ SEC4              YES
0101                *
0102   00EC CE 00F2 R SEC3   LDX #FORMAT           FORMAT ERROR
0103                         PRTMSG
0104 +  00EF 3F               SWI
0105 +  00F0 31               FCB 49
0106   00F1 39               RTS
0107                *
0108   00F2 20        FORMAT FCC ' FORMAT ERROR'
0109   00FF 0D               FCB $0D
0110                *
0111   0100 DE 20     SEC4   LDX DESCRA            POINT TO NAME
0112   0102 FF 002A R        STX BUFFER
0113   0105 96 22            LDA A DESCRC          GET LENGTH OF NAME
0114   0107 B7 002C R        STA A BUFFER+2
0115                         NXTOK                 GET TOKEN FROM CLI
0116 +  010A 3F               SWI
0117 +  010B 2F               FCB 47
0118   010C D6 25            LDA B RC              CHECK RC
0119   010E C1 2E            CMP B #'.             PERIOD?
0120   0110 26 DA            BNE SEC3              IF NOT, ERROR
0121                *
0122   0112 7C 002C R        INC BUFFER+2          COUNT PERIOD
0123                         NXTOK                 GET TOKEN FROM CLI
0124 +  0115 3F               SWI
0125 +  0116 2F               FCB 47
0126   0117 D6 25            LDA B RC              CHECK RC
0127   0119 C1 01            CMP B #1              UNAMBIG. NAME?
0128   011B 26 CF            BNE SEC3              IF NOT, ERROR
0129                *
0130   011D D6 22            LDA B DESCRC          GET LENGTH OF EXT
0131   011F FB 002C R        ADD B BUFFER+2        TOTAL LENGTH
0132   0122 CE 0010 R        LDX #SYSFCB+FCBNAM
0133                         PSHX                  POINTER TO FCBNAM
0134 +  0125 3F               SWI
0135 +  0126 05               FCB 5
0136   0127 FE 002A R        LDX BUFFER
0137                         PSHX                  POINTER TO CLI NAME
0138 +  012A 3F               SWI
0139 +  012B 05               FCB 5
0140                         FMTS                  FORMAT NAME INTO FCB
0141 +  012C 3F               SWI
0142 +  012D 34               FCB 52
0143   012E 31               INS
0144   012F 31               INS                   CLEAN STACK
0145   0130 31               INS
0146   0131 31               INS
0147   0132 5D               TST B                 ERRORS?
0148   0133 26 B7            BNE SEC3              YES
0149                *
0150   0135 CE 0000 R        LDX #SYSFCB
0151                         OPEND                 OPEN THE DIRECTORY
0152 +  0138 3F               SWI
0153 +  0139 17               FCB 23
0154   013A A6 05     SEC5   LDA A FCBSTA,X        CHECK STATUS
0155   013C 27 1D            BEQ SEC6              GOOD?
0156                *
0157   013E 81 01            CMP A #1              END OF DIRECTORY?
0158   0140 26 16            BNE SEC5A             NO
0159                *
0160   0142 CE 0148 R        LDX #FNFND            FILE NOT FOUND ON DISK
0161                         PRTMSG
0162 +  0145 3F               SWI
0163 +  0146 31               FCB 49
0164   0147 39               RTS
0165                *
0166   0148 20        FNFND  FCC ' FILE NOT FOUND'
0167   0157 0D               FCB $0D
0168                *
0169                  SEC5A  PRTERR                PRINT ERROR MESSAGE
0170 +  0158 3F               SWI
0171 +  0159 1E               FCB 30
0172   015A 39               RTS
0173                *
0174   015B EE 27     SEC6   LDX FCBIND,X          POINT TO DIRECTORY NAME
0175                         PSHX
0176 +  015D 3F               SWI
0177 +  015E 05               FCB 5
0178   015F CE 0010 R        LDX #SYSFCB+FCBNAM    POINT TO FCB NAME
0179                         PSHX
0180 +  0162 3F               SWI
0181 +  0163 05               FCB 5
0182   0164 C6 0C            LDA B #12             COMPARE 12 CHARACTERS
```

```
0183                            CMPC
0184 +  0166 3F                 SWI
0185 +  0167 12                 FCB 18
0186    0168 31                 INS
0187    0169 31                 INS                     CLEAN STACK
0188    016A 31                 INS
0189    016B 31                 INS
0190    016C 27 07              BEQ SEC7                FOUND ENTRY IN DIRECTORY?
0191                    *
0192    016E CE 0000 R          LDX #SYSFCB
0193                            GETDR                   GET NEW ENTRY
0194 +  0171 3F                 SWI
0195 +  0172 1A                 FCB 26
0196    0173 20 C5              BRA SEC5
0197                    *
0198                    SEC7    NXTOK                   GET TOKEN FROM CLI
0199 +  0175 3F                 SWI
0200 +  0176 2F                 FCB 47
0201    0177 96 26              LDA A CLASS
0202    0179 81 04              CMP A #4                DELIMITER?
0203    017B 27 03              BEQ SEC8                YES
0204                    *
0205    017D 7E 00EC R          JMP SEC3                NO, ERROR
0206                    *
0207                    SEC8    NXTOK                   GET TOKEN FROM CLI
0208 +  0180 3F                 SWI
0209 +  0181 2F                 FCB 47
0210    0182 D6 25              LDA B RC                CHECK RC
0211    0184 C1 03              CMP B #3                NUMBER?
0212    0186 27 03              BEQ SEC9                YES
0213                    *
0214    0188 7E 00C8 R  SEC8A   JMP SEC1                NO, ERROR
0215                    *
0216    018B 7D 0027    SEC9    TST VALUE               SECURITY-VALUE TOO BIG?
0217    018E 26 F8              BNE SEC8A               YES, ERROR
0218                    *
0219    0190 CE 0000 R          LDX #SYSFCB
0220    0193 EE 27              LDX FCBIND,X            POINT TO DIRECTORY ENTRY
0221    0195 96 28              LDA A VALUE+1           GET NEW ACCESS CODE
0222    0197 C6 0E              LDA B #FIBACS
0223                            ADDBX                   POINT TO AC. FIELD IN ENTRY
0224 +  0199 3F                 SWI
0225 +  019A 0A                 FCB 10
0226    019B A7 00              STA A 0,X               STORE IT
0227    019D CE 0000 R          LDX #SYSFCB
0228    01A0 63 06              COM FCBDTT,X            MAKE 'OUTPUT'
0229                            IOHDR                   WRITE DIRECTORY
0230 +  01A2 3F                 SWI
0231 +  01A3 13                 FCB 19
0232    01A4 6F 06              CLR FCBDTT,X            RESTORE 'INPUT'
0233    01A6 4D                 TST A                   ERROR?
0234    01A7 27 03              BEQ *+5                 NO
0235                    *
0236    01A9 7E 0158 R          JMP SEC5A               YES
0237                    *
0238    01AC 39                 RTS
0239                            END
```

```
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
BUFFER 002A R
CHAIN  243A M
CLASS  0026
CLOSE  2369 M
CMPC   231B M
CMWC   2572 M
CUCHAR 0023
DELETE 2420 M
DESCRA 0020
DESCRC 0022
DIV16  2524 M
FCBACS 001E
FCBBAK 000E
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBDTT 0006
FCBEQT 0000
FCBFTS 001F
FCBFWD 000C
FCBGDT 0002
FCBIND 0027
FCBLTS 0021
FCBNAM 0010
FCBNFB 0025
FCBNMS 0023
FCBSCF 0029
FCBSCT 000B
FCBSTA 0005
FCBTRK 000A
FCBTYP 001D
FIBACS 000E
FIBDEF 2940 M
FIBFTS 000F
FIBLTS 0011
FIBNAM 0000
FIBNMS 0013
FIBTYP 000D
FMTFCB 2488 M
FMTS   2558 M
FNFND  0148 R
FORMAT 00F2 R
GETDR  23EC M
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
IOHDR  2335 M
LOADB  246E M
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NUMBER 00CE R
NXTOK  24D6 M
OPEN   234F M
```

```
OPEND  239E M
PRTERR 2454 M
PRTMSG 250A M
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
RC     0025
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
SEC1   00C8 R
SEC1A  00DC R
SEC2   00E8 R
SEC3   00EC R
SEC4   0100 R
SEC5   013A R
SEC5A  0158 R
SEC6   015B R
SEC7   0175 R
SEC8   0180 R
SEC8A  0188 R
SEC9   018B R
SECSIZ 0080
SECURE 00AA R
SECURI 0000 RN
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
SYSFCB 0000 R
TABX   219C M
TXAB   2183 M
VALUE  0027
WRITE  23D2 M
XABX   21B5 M
```

```
0001    0000 0000     N       NAM SET
0002                          *
0003                          * SET COMMAND FOR CP/68
0004                          *
0005                          * SYNTAX:    SET DP=XX
0006                          *
0007                                BASEQU
0008 +  0000 0020             DESCRA EQU $20  DESCRIPTOR ADDRESS(2)
0009 +  0000 0022             DESCRC EQU $22  DESCRIPTOR COUNT
0010 +  0000 0023             CUCHAR EQU $23  CURRENT CHAR (2)
0011 +  0000 0025             RC     EQU $25  TOKEN RETURN CODE
0012 +  0000 0026             CLASS  EQU $26  TOKEN CLASS
0013 +  0000 0027             VALUE  EQU $27  BIN VALUE/TRANSFER ADDRESS (2)
0014 +  0000 0029             FCBCHN EQU $29  TOP OF FCB CHAIN (2)
0015 +  0000 002B             FRETAB EQU $2B  DISK FREE SPACE POINTER (8)
0016 +  0000 0033             BMEM   EQU $33  START OF TRANSIENT AREA(2)
0017 +  0000 0035             EMEM   EQU $35  END OF TRANSIENT AREA (2)
0018 +  0000 0037             CMEM   EQU $37  NEXT AVAIL TRANSIENT AREA (2)
0019 +  0000 0039             BS     EQU $39  BACKSPACE CHAR
0020 +  0000 003A             DL     EQU $3A  DELETE LINE CHAR
0021 +  0000 003B             DP     EQU $3B  DEPTH; LINES/PAGE
0022 +  0000 003C             DPCNT  EQU $3C  DEPTH TEMP
0023 +  0000 003D             WD     EQU $3D  WIDTH; CHARS/LINE
0024 +  0000 003E             NL     EQU $3E  NULL COUNT
0025 +  0000 003F             TB     EQU $3F  TAB CHAR
0026 +  0000 0040             DX     EQU $40  DUPLEX; FF=H, 00=F
0027 +  0000 0041             EJ     EQU $41  EJECT COUNT
0028 +  0000 0042             PS     EQU $42  PAUSE; 00=YES
0029 +  0000 0043             ES     EQU $43  ESCAPE CHAR
0030 +  0000 0044             LDP    EQU $44  DEPTH LINES/PAGE
0031 +  0000 0045             LDPCNT EQU $45  DEPTH TEMP
0032 +  0000 0046             LWD    EQU $46  WIDTH CHARS/LINE
0033                          *
0034                          SET0   NXTOK          GET TOKEN (PARM NAME)
0035 +  0000 3F                      SWI
0036 +  0001 2F                      FCB 47
0037    0002 DE 20                    LDX DESCRA    POINT TO TOKEN
0038    0004 A6 00                    LDA A 0,X
0039    0006 91 43                    CMP A ES      ESCAPE?
0040    0008 26 01                    BNE SET1      NO
0041                          *
0042    000A 39                       RTS           DONE RETURN TO CLI
0043                          *
0044    000B D6 25            SET1    LDA B RC      GET RETURN CODE
0045    000D C1 01                    CMP B #1      NAME?
0046    000F 27 08                    BEQ SET3      YES
0047                          *
0048    0011 CE 008D R        SET2    LDX #MSGA     ERROR
0049                                  PRTMSG
0050 +  0014 3F                       SWI
0051 +  0015 31                       FCB 49
0052    0016 7E 007F R                JMP SETNXT
0053                          *
0054    0019 DE 20            SET3    LDX DESCRA    GET PARM NAME
0055    001B EE 00                    LDX 0,X
0056                          *
0057                                  NXTOK         GET "="
0058 +  001D 3F                       SWI
0059 +  001E 2F                       FCB 47
0060    001F D6 25                    LDA B RC
0061    0021 C1 3D                    CMP B #'=     "="?
0062    0023 26 EC                    BNE SET2      ERROR
0063                          *
0064                                  NXTOK         GET VALUE
0065 +  0025 3F                       SWI
0066 +  0026 2F                       FCB 47
0067    0027 D6 25                    LDA B RC
0068    0029 C1 03                    CMP B #3      NUMBER?
0069    002B 27 42                    BEQ SET7      YES
0070                          *
0071    002D C1 01                    CMP B #1      NAME?
0072    002F 26 49                    BNE SET8      ERROR
0073                          *
0074    0031 8C 4458                  CPX #'DX      DUPLEX?
0075    0034 26 1A                    BNE SET5      NO
0076    0036 DE 20                    LDX DESCRA    YES
0077    0038 A6 00                    LDA A 0,X     GET RESPONSE
0078    003A 81 48                    CMP A #'H     HALF?
0079    003C 26 07                    BNE SET4      NO
0080                          *
0081    003E 86 FF                    LDA A #$FF
0082    0040 97 40                    STA A DX      SET TO HALF DUPLEX
0083    0042 7E 007F R                JMP SETNXT
0084                          *
0085    0045 81 46            SET4    CMP A #'F     FULL?
0086    0047 26 C8                    BNE SET2      ERROR
0087                          *
0088    0049 86 00                    LDA A #00
0089    004B 97 40                    STA A DX      DUPLEX=FULL
0090    004D 7E 007F R                JMP SETNXT
0091                          *
0092    0050 8C 5053          SET5    CPX #'PS      PAUSE?
0093    0053 26 25                    BNE SET8      NO
0094                          *
0095    0055 DE 20                    LDX DESCRA
0096    0057 A6 00                    LDA A 0,X     GET RESPONSE
0097    0059 81 4E                    CMP A #'N     "NO"?
0098    005B 26 07                    BNE SET6      NO
0099                          *
0100    005D 86 FF                    LDA A #$FF
0101    005F 97 42                    STA A PS      PAUSE OFF
0102    0061 7E 007F R                JMP SETNXT
0103                          *
0104    0064 81 59            SET6    CMP A #'Y     "YES"?
0105    0066 26 A9                    BNE SET2      ERROR
0106                          *
0107    0068 86 00                    LDA A #00
0108    006A 97 42                    STA A PS      PAUSE ON
0109    006C 7E 007F R                JMP SETNXT
0110                          *
0111                          * PARM=VALUE FOUND
0112                          *
0113    006F 8D 40            SET7    BSR SETSRC    LOOKUP
0114    0071 25 07                    BCS SET8      ERROR
0115                          *
0116                          * FOUND ENTRY X=A(PARM)
0117                          *
0118    0073 96 28                    LDA A VALUE+1 GET VALUE
0119    0075 A7 00                    STA A 0,X
0120    0077 7E 007F R                JMP SETNXT
0121                          *
```

```
0122   007A CE 009A R SET8   LDX #MSGB
0123                         PRTMSG
0124 +  007D 3F               SWI
0125 +  007E 31               FCB 49
0126   007F CE 00AB R SETNXT LDX #MSGC
0127                         PRTMSG
0128 +  0082 3F               SWI
0129 +  0083 31               FCB 49
0130                         GTCMD
0131 +  0084 3F               SWI
0132 +  0085 30               FCB 48
0133   0086 DE 20            LDX DESCRA
0134   0088 DF 23            STX CUCHAR
0135   008A 7E 0000 R        JMP SETO
0136                 *
0137   008D 53       MSGA    FCC 'SYNTAX ERROR'
0138   0099 0D               FCB $0D
0139                 *
0140   009A 49       MSGB    FCC 'INVALID SET PARM'
0141   00AA 0D               FCB $0D
0142                 *
0143   00AB 53       MSGC    FCC 'SET- '
0144   00B0 04               FCB $04
0145                 *
0146                 * SEARCH SETAB FOR AN ENTRY
0147                 *
0148                 SETSRC TXAB          PUT PARM NAME INTO A,B
0149 +  00B1 3F               SWI
0150 +  00B2 02               FCB 2
0151   00B3 CE 00CC R        LDX #SETAB    POINT TO TABLE
0152                 *
0153   00B6 A1 00    SETSR1  CMP A 0,X
0154   00B8 26 08            BNE SETSR2    NO MATCH
0155                 *
0156   00BA E1 01            CMP B 1,X
0157   00BC 26 04            BNE SETSR2    NO MATCH
0158                 *
0159                 * MATCH
0160                 *
0161   00BE EE 02            LDX 2,X       GET BP ADDRESS
0162   00C0 0C               CLC
0163   00C1 39               RTS
0164                 *
0165   00C2 08       SETSR2  INX           POINT TO NEXT ENTRY
0166   00C3 08               INX
0167   00C4 08               INX
0168   00C5 08               INX
0169   00C6 6D 00            TST 0,X       END OF TABLE?
0170   00C8 26 EC            BNE SETSR1    NO
0171                 *
0172   00CA 0D               SEC           NOT IN TABLE
0173   00CB 39               RTS
0174                 *
0175   00CC 42       SETAB   FCC 'BS'
0176   00CE 0039             FDB BS
0177                 *
0178   00D0 44               FCC 'DL'
0179   00D2 003A             FDB DL
0180                 *
0181   00D4 44               FCC 'DP'
0182   00D6 003B             FDB DP
0183                 *
0184   00D8 57               FCC 'WD'
0185   00DA 003D             FDB WD
0186                 *
0187   00DC 4E               FCC 'NL'
0188   00DE 003E             FDB NL
0189                 *
0190   00E0 54               FCC 'TB'
0191   00E2 003F             FDB TB
0192                 *
0193   00E4 45               FCC 'EJ'
0194   00E6 0041             FDB EJ
0195                 *
0196   00E8 45               FCC 'ES'
0197   00EA 0043             FDB ES
0198                 *
0199                 * LINE PRINTER SET PARMS
0200                 *
0201   00EC 4C               FCC 'LD'
0202   00EE 0044             FDB LDP
0203                 *
0204   00F0 4C               FCC 'LW'
0205   00F2 0046             FDB LWD
0206                 *
0207                         END
```

```
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
BMEM   0033
BS     0039
CHAIN  243A M
CLASS  0026
CLOSE  2369 M
CMEM   0037
CMPC   231B M
CMWC   2572 M
CUCHAR 0023
DELETE 2420 M
DESCRA 0020
DESCRC 0022
DIV16  2524 M
DL     003A
DP     003B
DPCNT  003C
DX     0040
EJ     0041
EMEM   0035
ES     0043
FCBCHN 0029
FCBDEF 2650 M
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
FRETAB 002B
GETDR  23EC M
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
IOHDR  2335 M
LDP    0044
LDPCNT 0045
LOADR  246E M
LWD    0046
MOVC   2301 M
MOVS   24A2 M
MSGA   008D R
MSGB   009A R
MSGC   00AB R
MUL16  22E7 M
MUL8   22CD M
NL     003E
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
PRTERR 2464 M
PRTMSG 250A M
PS     0042
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
RC     0025
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
SET    0000 RN
SET0   0000 R
SET1   000B R
SET2   0011 R
SET3   0019 R
SET4   0045 R
SET5   0050 R
SET6   0064 R
SET7   006F R
SET8   007A R
SETAB  00CC R
SETNXT 007F R
SETSR1 00B6 R
SETSR2 00C2 R
SETSRC 00B1 R
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
TABX   219C M
TB     003F
TXAB   2183 M
VALUE  0027
WD     003D
WRITE  23D2 M
XABX   21B5 M
```

```
0001    0000 0000     N        NAM STAT
0002                           *
0003                           * TRANSIENT TO LIST DEVICE ASSIGNMENTS
0004                           *
0005                           TABX           X:=A(PDTAB)
0006 +   0000 3F               SWI
0007 +   0001 03               FCB 3
0008    0002 FF 006E R         STX PDTAB
0009                           *
0010    0005 A6 00     STAT0   LDA A 0,X      END OF TABLE?
0011    0007 26 01             BNE *+3        NO
0012                           *
0013    0009 39                RTS
0014                           *
0015                           * MOVE DEV1,2 TO MSG
0016                           *
0017    000A B7 0064 R         STA A MSG
0018    000D B7 006A R         STA A MSG+6
0019    0010 A6 01             LDA A 1,X
0020    0012 B7 0065 R         STA A MSG+1
0021    0015 B7 006B R         STA A MSG+7
0022    0018 A6 02             LDA A 2,X
0023    001A B7 0066 R         STA A MSG+2
0024    001D B7 006C R         STA A MSG+8
0025                           *
0026                           * SEE IF ORIGINAL ASSIGNMENT
0027                           *
0028    0020 A6 03             LDA A 3,X
0029    0022 E6 04             LDA B 4,X
0030    0024 A1 05             CMP A 5,X
0031    0026 26 13             BNE STAT1      NO
0032                           *
0033    0028 E1 06             CMP B 6,X
0034    002A 26 0F             BNE STAT1      NO
0035                           *
0036                           * YES PRINT ASSIGNMENT
0037                           *
0038                           STAT0A PSHX
0039 +   002C 3F                SWI
0040 +   002D 05                FCB 5
0041    002E CE 0064 R         LDX #MSG
0042                           PRTMSG
0043 +   0031 3F                SWI
0044 +   0032 31                FCB 49
0045                           PULX
0046 +   0033 3F                SWI
0047 +   0034 06                FCB 6
0048    0035 C6 07             LDA B #7
0049                           ADDBX          POINT TO NEXT ENTRY
0050 +   0037 3F                SWI
0051 +   0038 0A                FCB 10
0052    0039 20 CA             BRA STAT0
0053                           *
0054                           *
0055                           STAT1  PSHX           SAVE POINTER
0056 +   003B 3F                SWI
0057 +   003C 05                FCB 5
0058    003D FE 006E R         LDX PDTAB
0059                           *
0060    0040 A1 05     STAT1A  CMP A 5,X
```

```
0061   0042 26 17           BNE STAT2     NO MATCH
0062                        *
0063   0044 E1 06           CMP B 6,X
0064   0046 26 13           BNE STAT2     NO MATCH
0065                        *
0066                        * FOUND ASSIGNMENT MOVE IN NAME
0067                        *
0068   0048 A6 00           LDA A 0,X
0069   004A B7 006A R       STA A MSG+6
0070   004D A6 01           LDA A 1,X
0071   004F B7 006B R       STA A MSG+7
0072   0052 A6 02           LDA A 2,X
0073   0054 B7 006C R       STA A MSG+8
0074                        PULX
0075 +  0057 3F              SWI
0076 +  0058 06              FCB 6
0077   0059 20 D1           BRA STAT0A
0078                        *
0079   005B 08              STAT2  INX           POINT TO NEXT ENTRY
0080   005C 08                     INX
0081   005D 08                     INX
0082   005E 08                     INX
0083   005F 08                     INX
0084   0060 08                     INX
0085   0061 08                     INX
0086   0062 20 DC           BRA STAT1A    TRY AGAIN
0087                        *
0088   0064 20              MSG    FCC '    =    '
0089   006D 0D                     FCB $0D
0090                        *
0091   006E 0002            PDTAB  RMB 2
0092                        *
0093                        END
```

```
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
CHAIN  243A M
CLOSE  2369 M
CMPC   231B M
CMWC   2572 M
DELETE 2420 M
DIV16  2524 M
FCBDEF 2650 M
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
GETDR  23EC M
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
IOHDR  2335 M
LOADB  246E M
MOVC   2301 M
MOVS   24A2 M
MSG    0064 R
MUL16  22E7 M
MUL8   22CD M
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
PDTAB  006E R
PRTERR 2454 M
PRTMSG 250A M
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
STAT   0000 RN
STAT0  0005 R
STAT0A 002C R
STAT1  003B R
STAT1A 0040 R
STAT2  005B R
SUBARX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
TABX   219C M
TXAB   2183 M
WRITE  23D2 M
XABX   21B5 M
```

```
0001   0000 0000    N          NAM RANDOM
0002                 *
0003                 * CP/68 RANDOM-ACCESS FILES PACKAGE
0004                 * COPYRIGHT 1979 BY HEMENWAY ASSOCIATES
0005                 *                  BOSTON, MASS.
0006                 *
0007   0000 00CB     N          ENT CREATE    BUILD A NEW RANDOM FILE
0008   0000 02E6     N          ENT ROPEN     OPEN A RANDOM FILE
0009   0000 0387     N          ENT RCLOSE    CLOSE A RANDOM FILE
0010   0000 03C2     N          ENT RREAD     READ A BYTE FROM RANDOM FILE
0011   0000 0406     N          ENT RWRITE    WRITE A BYTE TO RANDOM FILE
0012   0000 04D4     N          ENT POSITION  POSITION RANDOM FILE TO RECORD
0013   0000 05E5     N          ENT EXPAND    ADD RECORDS TO RANDOM FILE
0014                 *
0015                 * VECTORS TO INDIVIDUAL ROUTINES
0016                 *
0017   0000 7E 00CB R RNDVEC JMP CREATE
0018   0003 7E 02E6 R        JMP ROPEN
0019   0006 7E 0387 R        JMP RCLOSE
0020   0009 7E 03C2 R        JMP RREAD
0021   000C 7E 0406 R        JMP RWRITE
0022   000F 7E 04D4 R        JMP POSITION
0023   0012 7E 05E5 R        JMP EXPAND
0024                 *
0025                 * SET UP ADDRESSING EQUATES AND DEFINITIONS
0026                 *
0027                       BASEQU
0028 +   0015 0020       DESCRA EQU $20  DESCRIPTOR ADDRESS(2)
0029 +   0015 0022       DESCRC EQU $22  DESCRIPTOR COUNT
0030 +   0015 0023       CUCHAR EQU $23  CURRENT CHAR (2)
0031 +   0015 0025       RC     EQU $25  TOKEN RETURN CODE
0032 +   0015 0026       CLASS  EQU $26  TOKEN CLASS
0033 +   0015 0027       VALUE  EQU $27  BIN VALUE/TRANSFER ADDRESS (2)
0034 +   0015 0029       FCBCHN EQU $29  TOP OF FCB CHAIN (2)
0035 +   0015 002B       FRETAB EQU $2B  DISK FREE SPACE POINTER (8)
0036 +   0015 0033       BMEM   EQU $33  START OF TRANSIENT AREA(2)
0037 +   0015 0035       EMEM   EQU $35  END OF TRANSIENT AREA (2)
0038 +   0015 0037       CMEM   EQU $37  NEXT AVAIL TRANSIENT AREA (2)
0039 +   0015 0039       BS     EQU $39  BACKSPACE CHAR
0040 +   0015 003A       DL     EQU $3A  DELETE LINE CHAR
0041 +   0015 003B       DP     EQU $3B  DEPTH; LINES/PAGE
0042 +   0015 003C       DPCNT  EQU $3C  DEPTH TEMP
0043 +   0015 003D       WD     EQU $3D  WIDTH; CHARS/LINE
0044 +   0015 003E       NL     EQU $3E  NULL COUNT
0045 +   0015 003F       TB     EQU $3F  TAB CHAR
0046 +   0015 0040       DX     EQU $40  DUPLEX; FF=H, 00=F
0047 +   0015 0041       EJ     EQU $41  EJECT COUNT
0048 +   0015 0042       PS     EQU $42  PAUSE; 00=YES
0049 +   0015 0043       ES     EQU $43  ESCAPE CHAR
0050 +   0015 0044       LDP    EQU $44  DEPTH LINES/PAGE
0051 +   0015 0045       LDPCNT EQU $45  DEPTH TEMP
0052 +   0015 0046       LWD    EQU $46  WIDTH CHARS/LINE
0053                       FCBDEF
0054 +   0015 0000       FCBEQT EQU 0         EQUIPMENT TABLE ADDRESS
0055 +   0015 0002       FCBGDT EQU 2         GENERIC DEVICE TYPE
0056 +   0015 0005       FCBSTA EQU 5         STATUS
0057 +   0015 0006       FCBDTT EQU 6         DATA TRANSFER TYPE
0058 +   0015 0007       FCBDBA EQU 7         DATA BUFFER ADDRESS
0059 +   0015 0009       FCBDRV EQU 9         DRIVE NUMBER
0060 +   0015 000A       FCBTRK EQU 10        TRACK NUMBER
0061 +   0015 000B       FCBSCT EQU 11        SECTOR NUMBER
0062 +   0015 000C       FCBFWD EQU 12        FWD LINK TRACK/SECTOR
0063 +   0015 000E       FCBBAK EQU 14        BACK LINK TRACK/SECTOR
0064 +   0015 0010       FCBNAM EQU 16        FILE NAME (8.3+EOT=13))
0065 +   0015 001D       FCBTYP EQU 29        FILE TYPE
0066 +   0015 001E       FCBACS EQU 30        FILE ACCESS CODE
0067 +   0015 001F       FCBFTS EQU 31        FIRST TRACK/SECTOR
0068 +   0015 0021       FCBLTS EQU 33        LAST TRACK/SECTOR
0069 +   0015 0023       FCBNMS EQU 35        NUMBER OF SECTORS
0070 +   0015 0025       FCBNFB EQU 37        NEXT FCB IN ACTIVE CHAIN
0071 +   0015 0027       FCBIND EQU 39        INDEX INTO DATA BUFFER
0072 +   0015 0029       FCBSCF EQU 41        SPACE COMPRESSION FLAG
0073                 *
0074                 * DISK ATTRIBUTE SECTION
0075                 *
0076   0015 001A     TRKSIZ EQU  26       NUMBER OF SECTORS IN TRACK
0077   0015 0080     SECSIZ EQU 128       NUMBER OF BYTES IN SECTOR
0078   0015 0080     BUFSIZ EQU SECSIZ
0079   0017 09B0     MXRNUM EQU SECSIZ-4*20  MAX. NO. OF RECORDS IN FILE
0080   0017 0003     FRESEC EQU 3         FREE-SPACE SECTOR ON TRACK 0
0081                 *
0082   0017 002A     FCBRNM EQU 42        NO. OF RECORDS IN FILE (FCB POINTERS)
0083   0017 002C     FCBRSZ EQU 44        NO. OF BYTES IN RECORD
0084   0017 002E     FCBRCD EQU 46        PRESENT RECORD NUMBER
0085   0017 0030     FCBPOS EQU 48        PRESENT POSITION IN RECORD
0086   0017 0032     FCBRTB EQU 50        START OF INDEXING TABLE
0087   0017 00AA     FCBRTE EQU 170       END OF INDEXING TABLE
0088                 *
0089                 * LOCAL FILE-CONTROL-BLOCK
0090                 *
0091   0017 0002     RNDFCB RMB 2
0092   0019 44              FCC 'DSK'
0093   001C 0002            RMB 2
0094   001E 0041   R        FDB RNDBUF
0095   0020 0021            RMB 33
0096   0041 0080     RNDBUF RMB SECSIZ    LOCAL SECTOR BUFFER
0097                 *
0098   00C1 0002     RNMTMP RMB 2         TEMP. LOCATIONS
0099   00C3 0002     RSZTMP RMB 2
0100   00C5 0002     RINTMP RMB 2
0101   00C7 0001     TMPTRK RMB 1
0102   00C8 0001     TMPSCT RMB 1
0103   00C9 0001     TMPPNT RMB 1
0104   00CA 0001     SAVEA  RMB 1
0105                 *
0106                 *
0107                 * RANDOM-FILE ERROR NUMBERS
0108                 *
0109                 * 11. =BAD RECORD-SIZE PARAMETER
0110                 * 12. =BAD NO. OF RECORDS PARAMETER
0111                 * 14. =BAD FILE TYPE (NOT RANDOM=02)
0112                 * 15. =POSITION PARAMETER OUTSIDE FILE
0113                 *
0115                 * CREATE A NEW RANDOM-ACCESS FILE
0116                 * CALL WITH FCB-ADDRESS IN INDEX REGISTER
0117                 * FCB MUST HAVE RANDOM-FILE EXTENSION
0118                 * (170 BYTES IN LENGTH)
0119                 * SET DRIVE, FILENAME, ACCESS CODE,
0120                 * NO. OF RECORDS, RECORD SIZE
0121                 * FILE TYPE WILL BE 02
0122                 *
```

```
0123                    CREATE PSHX             SAVE FCB ADDRESS
0124 +  00CB 3F                 SWI
0125 +  00CC 05                 FCB 5
0126    00CD 6F 05              CLR FCBSTA,X    INIT. STATUS
0127    00CF 6F 29              CLR FCBSCF,X    COMPRESSION OFF
0128    00D1 6F 1E              CLR FCBACS,X    ACCESS CODE=00
0129    00D3 86 02              LDA A #2
0130    00D5 A7 1D              STA A FCBTYP,X  TYPE=02 (RANDOM)
0131    00D7 6D 2C              TST FCBRSZ,X    CHECK RECORD SIZE>0
0132    00D9 26 10              BNE CR2
0133                    *
0134    00DB 6D 2D              TST FCBRSZ+1,X
0135    00DD 26 0C              BNE CR2
0136                    *
0137    00DF 86 0B              LDA A #11       ERROR NO. 11
0138                    * HANDLE ERROR RETURNS HERE
0139                    *
0140    00E1 CE 0017 R  CRERR   LDX #RNDFCB
0141                            CLOSE           BE SURE LOCAL FCB IS CLOSED
0142 +  00E4 3F                 SWI
0143 +  00E5 15                 FCB 21
0144                            PULX            RECOVER FCB ADDRESS
0145 +  00E6 3F                 SWI
0146 +  00E7 06                 FCB 6
0147    00E8 A7 05              STA A FCBSTA,X PUT IN ERROR STATUS
0148    00EA 39                 RTS
0149                    *
0150    00EB EE 2A      CR2     LDX FCBRNM,X    CHECK RECORD NUMBER
0151    00ED 27 09              BEQ CR3         MUST BE >0
0152                    *
0153                            TXAB
0154 +  00EF 3F                 SWI
0155 +  00F0 02                 FCB 2
0156    00F1 CE 09B0            LDX #MXRNUM     MUST BE <MXRNUM
0157                            SUBABX
0158 +  00F4 3F                 SWI
0159 +  00F5 0C                 FCB 12
0160    00F6 2A 04              BPL CR4
0161                    *
0162    00F8 86 0C      CR3     LDA A #12       ERROR NO. 12
0163    00FA 20 E5              BRA CRERR
0164                    *
0165    00FC 30         CR4     TSX
0166    00FD EE 00              LDX 0,X         POINT TO FCB
0167    00FF 86 FF              LDA A #$FF
0168    0101 A7 06              STA A FCBDTT,X OUTPUT
0169                            OPEN            OPEN FILE
0170 +  0103 3F                 SWI
0171 +  0104 14                 FCB 20
0172    0105 A6 05              LDA A FCBSTA,X ERROR?
0173    0107 26 D8              BNE CRERR       YES
0174                    *
0175    0109 A6 2A      CR5     LDA A FCBRNM,X GET NO. OF RECORDS
0176    010B E6 2B              LDA B FCBRNM+1,X
0177                    *
0178                    * FIRST FOUR BYTES OF FILE=NO. OF RECORDS, SIZE
0179                    *
0180                            WRITE           WRITE NO. OF RECORDS
0181 +  010D 3F                 SWI
0182 +  010E 19                 FCB 25
0183    010F A6 05              LDA A FCBSTA,X ERROR?
0184    0111 26 CE              BNE CRERR       YES
0185                    *
0186    0113 17                 TBA
0187                            WRITE
0188 +  0114 3F                 SWI
0189 +  0115 19                 FCB 25
0190    0116 A6 05              LDA A FCBSTA,X ERROR?
0191    0118 26 C7              BNE CRERR       YES
0192                    *
0193    011A A6 2C              LDA A FCBRSZ,X GET RECORD SIZE
0194    011C E6 2D              LDA B FCBRSZ+1,X
0195                            WRITE           WRITE RECORD SIZE
0196 +  011E 3F                 SWI
0197 +  011F 19                 FCB 25
0198    0120 A6 05              LDA A FCBSTA,X ERROR?
0199    0122 26 BD              BNE CRERR       YES
0200                    *
0201    0124 17                 TBA
0202                            WRITE
0203 +  0125 3F                 SWI
0204 +  0126 19                 FCB 25
0205    0127 A6 05              LDA A FCBSTA,X ERROR?
0206    0129 26 B6              BNE CRERR       YES
0207                    *
0208    012B CE 0017 R          LDX #RNDFCB
0209    012E 6F 05              CLR FCBSTA,X    INIT. STATUS
0210    0130 6F 29              CLR FCBSCF,X    NO COMPRESSION
0211    0132 86 FF              LDA A #$FF
0212    0134 A7 06              STA A FCBDTT,X OUTPUT
0213    0136 86 20              LDA A #$20
0214    0138 A7 10              STA A FCBNAM,X NAME HAS BLANK IN IT
0215    013A 30                 TSX
0216    013B EE 00              LDX 0,X         POINT TO FCB
0217    013D A6 09              LDA A FCBDRV,X GET DRIVE NUMBER
0218    013F CE 0017 R          LDX #RNDFCB
0219    0142 A7 09              STA A FCBDRV,X SAVE IT IN LOCAL FCB
0220                    *
0221                    * NOW ALLOCATE SPACE ON DISK FOR FILE
0222                    * BUILD INDEX OF RECORD POINTERS USING USER FCB
0223                    * DATA SPACE BUILT USING RNDFCB
0224                    *
0225                    * RECORD POINTER HAS THREE BYTES:
0226                    * 1. TRACK
0227                    * 2. SECTOR
0228                    * 3. POINTER = FCBIND-FCBDBA
0229                    *
0230                            OPEN            OPEN LOCAL FILE (DATA)
0231 +  0144 3F                 SWI
0232 +  0145 14                 FCB 20
0233    0146 A6 05              LDA A FCBSTA,X  ERROR?
0234    0148 27 03              BEQ CR6         NO
0235                    *
0236    014A 7E 00E1 R  CR5A    JMP CRERR       YES
0237                    *
0238    014D CE 0017 R  CR6     LDX #RNDFCB
0239    0150 A6 1F              LDA A FCBFTS,X  GET FIRST TRACK
0240    0152 E6 20              LDA B FCBFTS+1,X    FIRST SECTOR
0241    0154 30                 TSX
0242    0155 EE 00              LDX 0,X         POINT TO FCB
0243                            WRITE           WRITE TRACK
0244 +  0157 3F                 SWI
```

```
0245 +  0158 19              FCB 25
0246    0159 A6 05           LDA A FCBSTA,X  ERROR?
0247    015B 26 ED           BNE CR5A        YES
0248                 *
0249    015D 17              TBA
0250                         WRITE           WRITE SECTOR
0251 +  015E 3F              SWI
0252 +  015F 19              FCB 25
0253    0160 A6 05           LDA A FCBSTA,X  ERROR?
0254    0162 26 E6           BNE CR5A        YES
0255                 *
0256    0164 86 04           LDA A #4        FIRST POINTER =4
0257                         WRITE           WRITE POINTER
0258 +  0166 3F              SWI
0259 +  0167 19              FCB 25
0260    0168 A6 05           LDA A FCBSTA,X  ERROR?
0261    016A 26 DE           BNE CR5A        YES
0262                 *
0263    016C A6 2A           LDA A FCBRNM,X
0264    016E E6 2B           LDA B FCBRNM+1,X
0265    0170 B7 00C1 R       STA A RNMTMP    INIT. TEMP. RECNUM
0266    0173 F7 00C2 R       STA B RNMTMP+1
0267                 *
0268                 * LOOP THROUGH HERE FOR EACH RECORD
0269                 *
0270    0176 A6 2C   CR7     LDA A FCBRSZ,X
0271    0178 E6 2D           LDA B FCBRSZ+1,X
0272    017A B7 00C3 R       STA A RSZTMP    INIT. TEMP. RECSIZ
0273    017D F7 00C4 R       STA B RSZTMP+1
0274                 *
0275                 * WRITE OUT ONE RECORD OF NULLS
0276                 *
0277    0180 CE 0017 R CR7A  LDX #RNDFCB
0278    0183 4F              CLR A
0279                         WRITE
0280 +  0184 3F              SWI
0281 +  0185 19              FCB 25
0282    0186 A6 05           LDA A FCBSTA,X  ERROR?
0283    0188 26 C0           BNE CR5A        YES
0284                 *
0285    018A FE 00C3 R       LDX RSZTMP
0286    018D 09              DEX             COUNT DOWN RECORD
0287    018E FF 00C3 R       STX RSZTMP
0288    0191 26 ED           BNE CR7A        LOOP UNTIL RECORD DONE
0289                 *
0290    0193 FE 00C1 R       LDX RNMTMP
0291    0196 09              DEX             COUNT DOWN NO. OF RECORDS
0292    0197 FF 00C1 R       STX RNMTMP
0293    019A 27 2D           BEQ CR8         DONE?
0294                 *
0295                 * OUTPUT INDEX BLOCK HERE
0296                 *
0297    019C CE 0017 R       LDX #RNDFCB
0298    019F A6 0A           LDA A FCBTRK,X  GET TRACK
0299    01A1 E6 0B           LDA B FCBSCT,X  GET SECTOR
0300    01A3 30              TSX
0301    01A4 EE 00           LDX 0,X         POINT TO FCB
0302                         WRITE           WRITE TRACK
0303 +  01A6 3F              SWI
0304 +  01A7 19              FCB 25
0305    01A8 A6 05           LDA A FCBSTA,X  ERROR?
0306    01AA 27 03           BEQ CR7C        NO
0307                 *
0308    01AC 7E 00E1 R CR7B  JMP CRERR       YES
0309                 *
0310    01AF 17      CR7C    TBA
0311                         WRITE           WRITE SECTOR
0312 +  01B0 3F              SWI
0313 +  01B1 19              FCB 25
0314    01B2 A6 05           LDA A FCBSTA,X  ERROR?
0315    01B4 26 F6           BNE CR7B        YES
0316                 *
0317    01B6 CE 0017 R       LDX #RNDFCB
0318    01B9 A6 28           LDA A FCBIND+1,X
0319    01BB A0 08           SUB A FCBDBA+1,X  FIND POINTER
0320    01BD 30              TSX
0321    01BE EE 00           LDX 0,X         POINT TO FCB
0322                         WRITE           WRITE POINTER
0323 +  01C0 3F              SWI
0324 +  01C1 19              FCB 25
0325    01C2 A6 05           LDA A FCBSTA,X  ERROR?
0326    01C4 26 E6           BNE CR7B        YES
0327                 *
0328    01C6 7E 0176 R       JMP CR7         LOOP ON NO. OF RECORDS
0329                 *
0330                 * NOW HAVE INDEX FILE AND DATA FILE
0331                 * APPEND DATA FILE TO INDEX FILE
0332                 *
0333    01C9 CE 0017 R CR8   LDX #RNDFCB
0334                 *
0335                 * CLOSE DATA FILE NOW
0336                 * DUPLICATE MOST OF 'CLOSE' ROUTINE IN CP/68
0337                 * DOES NOT UPDATE DIRECTORY
0338                 *
0339                         TXAB            FCB ADDRESS IN (A,B)
0340 +  01CC 3F              SWI
0341 +  01CD 02              FCB 2
0342    01CE DE 29           LDX FCBCHN      GET HEAD OF FCB-CHAIN
0343                         PSHX            SAVE X
0344 +  01D0 3F              SWI
0345 +  01D1 05              FCB 5
0346                         SUBABX          AT THIS FCB?
0347 +  01D2 3F              SWI
0348 +  01D3 0C              FCB 12
0349                         PULX            RESTORE X
0350 +  01D4 3F              SWI
0351 +  01D5 06              FCB 6
0352    01D6 26 0A           BNE CR8A        NOT HERE YET
0353                 *
0354    01D8 A6 25           LDA A FCBNFB,X
0355    01DA E6 26           LDA B FCBNFB+1,X
0356    01DC 97 29           STA A FCBCHN    MAKE NEW HEAD FCB
0357    01DE D7 2A           STA B FCBCHN+1
0358    01E0 20 1C           BRA CR8C
0359                 *
0360    01E2 A1 25   CR8A    CMP A FCBNFB,X AT DESIRED FCB?
0361    01E4 26 14           BNE CR8B        NO
0362                 *
0363    01E6 E1 26           CMP B FCBNFB+1,X AT DESIRED FCB?
0364    01E8 26 10           BNE CR8B         NO
0365                 *
0366                 * FIX FCB-CHAIN TO GO AROUND THIS FCB
```

```
0367                    *
0368                            PSHX            SAVE X
0369 +  01EA 3F                 SWI
0370 +  01EB 05                 FCB 5
0371                            TABX            POINT TO THIS FCB
0372 +  01EC 3F                 SWI
0373 +  01ED 03                 FCB 3
0374    01EE A6 25              LDA A FCBNFB,X
0375    01F0 E6 26              LDA B FCBNFB+1,X
0376                            PULX
0377 +  01F2 3F                 SWI
0378 +  01F3 06                 FCB 6
0379    01F4 A7 25              STA A FCBNFB,X  UPDATE LINKAGE AROUND FCB
0380    01F6 E7 26              STA B FCBNFB+1,X
0381    01F8 20 04              BRA CR8C
0382                    *
0383    01FA EE 25      CR8B    LDX FCBNFB,X    GET NEXT FCB IN CHAIN
0384    01FC 20 E4              BRA CR8A        KEEP LOOKING FOR FCB
0385                    *
0386    01FE CE 0017 R  CR8C    LDX #RNDFCB     POINT TO DATA FCB
0387                    *
0388                    * WRITE OUT LAST SECTOR OF DATA
0389                    *
0390    0201 6D 0A              TST FCBTRK,X    AT END OF DISK?
0391    0203 27 04              BEQ CR8D        YES
0392                    *
0393    0205 6D 0B              TST FCBSCT,X    AT END OF DISK?
0394    0207 26 0A              BNE CR8E        NO
0395                    *
0396    0209 A6 0E      CR8D    LDA A FCBBAK,X  FIXUP FOR END-OF-DISK
0397    020B E6 0F              LDA B FCBBAK+1,X
0398    020D A7 0A              STA A FCBTRK,X
0399    020F E7 0B              STA B FCBSCT,X
0400    0211 20 12              BRA CR8F
0401                    *
0402                    CR8E    IOHDR           WRITE LAST DATA SECTOR
0403 +  0213 3F                 SWI
0404 +  0214 13                 FCB 19
0405    0215 A6 05              LDA A FCBSTA,X  ERROR?
0406    0217 26 93              BNE CR7B        YES
0407                    *
0408    0219 A6 23              LDA A FCBNMS,X  ONE MORE SECTOR IN COUNT
0409    021B E6 24              LDA B FCBNMS+1,X
0410    021D CB 01              ADD B #1
0411    021F 89 00              ADC A #0
0412    0221 A7 23              STA A FCBNMS,X
0413    0223 E7 24              STA B FCBNMS+1,X
0414    0225 A6 0A      CR8F    LDA A FCBTRK,X    GET LAST TRACK WRITTEN
0415    0227 E6 0B              LDA B FCBSCT,X              SECTOR
0416    0229 A7 21              STA A FCBLTS,X    UPDATE LT,LS
0417    022B E7 22              STA B FCBLTS+1,X
0418    022D 6F 06              CLR FCBDTT,X      MAKE "INPUT"
0419    022F 86 00              LDA A #0          TRACK=0
0420    0231 C6 03              LDA B #FRESEC     FREE-SPACE SECTOR
0421    0233 A7 0A              STA A FCBTRK,X
0422    0235 E7 0B              STA B FCBSCT,X
0423                            IOHDR             READ FREE-SPACE SECTOR
0424 +  0237 3F                 SWI
0425 +  0238 13                 FCB 19
0426    0239 A6 05              LDA A FCBSTA,X    ERROR?
0427    023B 26 62              BNE CR9A          YES
```

```
0428                    *
0429    023D 63 06              COM FCBDTT,X      MAKE "OUTPUT"
0430    023F A6 09              LDA A FCBDRV,X    GET DRIVE NO.
0431    0241 84 03              AND A #$03        LIMIT RANGE (0-3)
0432    0243 48                 ASL A             2 BYTES/TABLE ENTRY
0433    0244 CE 002B            LDX #FRETAB
0434                            ADDAX             ACCESS FREE-SPACE SECTOR
0435 +  0247 3F                 SWI
0436 +  0248 09                 FCB 9
0437    0249 A6 00              LDA A 0,X         GET FREE TRACK
0438    024B E6 01              LDA B 1,X         GET FREE SECTOR
0439    024D CE 0041 R          LDX #RNDBUF       POINT TO SECTOR BUFFER
0440    0250 A7 7E              STA A SECSIZ-2,X  PUT NEW T/S INTO BUFFER
0441    0252 E7 7F              STA B SECSIZ-1,X
0442    0254 CE 0017 R          LDX #RNDFCB       POINT TO DATA FCB
0443                            IOHDR             WRITE OUT UPDATED FREE-SPACE SECTO
0444 +  0257 3F                 SWI
0445 +  0258 13                 FCB 19
0446    0259 A6 05              LDA A FCBSTA,X    ERROR?
0447    025B 26 42              BNE CR9A          YES
0448                    *
0449    025D 30         CR9     TSX
0450    025E EE 00              LDX 0,X         POINT TO FCB
0451    0260 4F                 CLR A
0452                    *
0453                    * LAST INDEX BLOCK=0,0,0
0454                    *
0455                            WRITE
0456 +  0261 3F                 SWI
0457 +  0262 19                 FCB 25
0458    0263 A6 05              LDA A FCBSTA,X  ERROR?
0459    0265 26 38              BNE CR9A        YES
0460                    *
0461                            WRITE
0462 +  0267 3F                 SWI
0463 +  0268 19                 FCB 25
0464    0269 A6 05              LDA A FCBSTA,X  ERROR?
0465    026B 26 32              BNE CR9A        YES
0466                    *
0467                            WRITE
0468 +  026D 3F                 SWI
0469 +  026E 19                 FCB 25
0470    026F A6 05              LDA A FCBSTA,X  ERROR?
0471    0271 26 2C              BNE CR9A        YES
0472                    *
0473    0273 CE 0017 R          LDX #RNDFCB
0474    0276 A6 1F              LDA A FCBFTS,X  GET FIRST T/S
0475    0278 E6 20              LDA B FCBFTS+1,X
0476    027A 30                 TSX
0477    027B EE 00              LDX 0,X         POINT TO FCB
0478    027D EE 07              LDX FCBDBA,X    POINT TO BUFFER
0479    027F A7 00              STA A 0,X       UPDATE FORWARD LINKS
0480    0281 E7 01              STA B 1,X
0481    0283 30                 TSX
0482    0284 EE 00              LDX 0,X         POINT TO FCB
0483    0286 A6 0A              LDA A FCBTRK,X  GET LAST T/S OF INDEX FILE
0484    0288 E6 0B              LDA B FCBSCT,X
0485    028A CE 0017 R          LDX #RNDFCB
0486    028D 36                 PSH A
0487    028E A6 1F              LDA A FCBFTS,X  POINT LOCAL FCB TO FIRST T/S
0488    0290 A7 0A              STA A FCBTRK,X
```

```
0489    0292 A6 20           LDA A FCBFTS+1,X
0490    0294 A7 0B           STA A FCBSCT,X
0491    0296 6F 06           CLR FCBDTT,X     INPUT
0492                         IOHDR            READ SECTOR
0493 +  0298 3F              SWI
0494 +  0299 13              FCB 19
0495    029A A6 05           LDA A FCBSTA,X   ERROR?
0496    029C 27 04           BEQ CR9B         NO
0497                 *
0498    029E 31              INS              CLEAN STACK
0499    029F 7E 00E1 R CR9A  JMP CRERR        YES
0500                 *
0501    02A2 63 06   CR9B    COM FCBDTT,X     OUTPUT
0502    02A4 32              PUL A
0503    02A5 EE 07           LDX FCBDBA,X     POINT TO BUFFER
0504    02A7 A7 02           STA A 2,X        UPDATE BACK LINKS
0505    02A9 E7 03           STA B 3,X
0506    02AB CE 0017 R       LDX #RNDFCB
0507                         IOHDR            WRITE SECTOR
0508 +  02AE 3F              SWI
0509 +  02AF 13              FCB 19
0510    02B0 A6 05           LDA A FCBSTA,X   ERROR?
0511    02B2 26 EB           BNE CR9A         YES
0512                 *
0513    02B4 A6 23           LDA A FCBNMS,X   GET NO. OF SECTORS
0514    02B6 E6 24           LDA B FCBNMS+1,X
0515    02B8 30              TSX
0516    02B9 EE 00           LDX 0,X          POINT TO FCB
0517    02BB EB 24           ADD B FCBNMS+1,X
0518    02BD A9 23           ADC A FCBNMS,X   ADD UP NOS.
0519    02BF A7 23           STA A FCBNMS,X
0520    02C1 E7 24           STA B FCBNMS+1,X
0521                         IOHDR            WRITE LAST INDEX SECTOR
0522 +  02C3 3F              SWI
0523 +  02C4 13              FCB 19
0524    02C5 A6 05           LDA A FCBSTA,X   ERROR?
0525    02C7 26 D6           BNE CR9A         YES
0526                 *
0527    02C9 CE 0017 R       LDX #RNDFCB
0528    02CC A6 21           LDA A FCBLTS,X
0529    02CE E6 22           LDA B FCBLTS+1,X GET LAST T/S OF DATA FILE
0530    02D0 30              TSX
0531    02D1 EE 00           LDX 0,X          POINT TO FCB
0532    02D3 6F 06           CLR FCBDTT,X     INPUT
0533    02D5 A7 0A           STA A FCBTRK,X
0534    02D7 E7 0B           STA B FCBSCT,X
0535                         IOHDR            READ SECTOR
0536 +  02D9 3F              SWI
0537 +  02DA 13              FCB 19
0538    02DB A6 05           LDA A FCBSTA,X   ERROR?
0539    02DD 26 C0           BNE CR9A         YES
0540                 *
0541    02DF 63 06           COM FCBDTT,X     OUTPUT
0542                         CLOSE            CLOSE FILE
0543 +  02E1 3F              SWI
0544 +  02E2 15              FCB 21
0545                         PULX             CLEAN STACK
0546 +  02E3 3F              SWI
0547 +  02E4 06              FCB 6
0548    02E5 39              RTS              DONE!
0549                 *
```

```
0551                 * OPEN A RANDOM-ACCESS FILE
0552                 * CALL WITH ADDRESS OF FCB IN INDEX REGISTER
0553                 * MUST HAVE EXTENDED FCB (170 BYTES)
0554                 * SET DRIVE, FILENAME
0555                 *
0556    ROPEN  PSHX              SAVE FCB ADDRESS
0557 +  02E6 3F              SWI
0558 +  02E7 05              FCB 5
0559    02E8 6F 05           CLR FCBSTA,X     INIT. STATUS
0560    02EA 6F 29           CLR FCBSCF,X     NO COMPRESSION
0561    02EC 6F 06           CLR FCBDTT,X     INPUT
0562                         OPEN             OPEN FILE
0563 +  02EE 3F              SWI
0564 +  02EF 14              FCB 20
0565    02F0 A6 05           LDA A FCBSTA,X ERROR?
0566    02F2 27 05           BEQ ROP2         NO
0567                 *
0568    ROPERR PULX              RECOVER FCB ADDRESS
0569 +  02F4 3F              SWI
0570 +  02F5 06              FCB 6
0571    02F6 A7 05           STA A FCBSTA,X SET STATUS
0572    02F8 39              RTS
0573                 *
0574    02F9 A6 1D   ROP2    LDA A FCBTYP,X CHECK TYPE OF FILE
0575    02FB 81 02           CMP A #2         RANDOM?
0576    02FD 27 06           BEQ ROP3         YES
0577                 *
0578    02FF 86 0E           LDA A #14        ERROR NUMBER 14
0579    ROP2A  CLOSE             CLOSE FILE
0580 +  0301 3F              SWI
0581 +  0302 15              FCB 21
0582    0303 20 EF           BRA ROPERR
0583                 *
0584    ROP3   READ              READ NUMBER OF RECORDS FROM FILE
0585 +  0305 3F              SWI
0586 +  0306 18              FCB 24
0587    0307 E6 05           LDA B FCBSTA,X ERROR?
0588    0309 27 03           BEQ ROP3B        NO
0589                 *
0590    030B 17      ROP3A   TBA
0591    030C 20 F3           BRA ROP2A        YES
0592                 *
0593    030E A7 2A   ROP3B   STA A FCBRNM,X
0594                         READ
0595 +  0310 3F              SWI
0596 +  0311 18              FCB 24
0597    0312 E6 05           LDA B FCBSTA,X ERROR?
0598    0314 26 F5           BNE ROP3A        YES
0599                 *
0600    0316 A7 2B           STA A FCBRNM+1,X
0601                         READ             READ RECORD SIZE FROM FILE
0602 +  0318 3F              SWI
0603 +  0319 18              FCB 24
0604    031A E6 05           LDA B FCBSTA,X ERROR?
0605    031C 26 ED           BNE ROP3A        YES
0606                 *
0607    031E A7 2C           STA A FCBRSZ,X
0608                         READ
0609 +  0320 3F              SWI
0610 +  0321 18              FCB 24
0611    0322 E6 05           LDA B FCBSTA,X ERROR?
```

```
0612   0324 26 E5            BNE ROP3A       YES
0613                  *
0614   0326 A7 2D            STA A FCBRSZ+1,X
0615                  *
0616                  * NOW CLEAR INDEXING TABLE IN FCB
0617                  *
0618   0328 C6 78            LDA B #FCBRTE-FCBRTB TOTAL LENGTH OF TABLE
0619   032A 6F 32      ROP4  CLR FCBRTB,X          CLEAR A BYTE
0620   032C 08               INX
0621   032D 5A               DEC B
0622   032E 26 FA            BNE ROP4              LOOP UNTIL DONE
0623                  *
0624   0330 30               TSX
0625   0331 EE 00            LDX 0,X         RESTORE FCB POINTER
0626   0333 86 32            LDA A #FCBRTB
0627                         ADDAX           POINT TO TABLE
0628 +  0335 3F               SWI
0629 +  0336 09               FCB 9
0630   0337 FF 00C5 R        STX RINTMP      SAVE TEMP. POINTER
0631                  *
0632                  * NOW READ IN INDEX AND BUILD FCB TABLE
0633                  *
0634   033A 30               TSX
0635   033B EE 00            LDX 0,X         POINT TO FCB
0636   033D A6 0A            LDA A FCBTRK,X
0637   033F E6 0B            LDA B FCBSCT,X
0638   0341 FE 00C5 R ROP4A  LDX RINTMP      POINT TO TABLE IN FCB
0639   0344 A7 00            STA A 0,X       PUT IN NEW ENTRY (T/S)
0640   0346 E7 01            STA B 1,X
0641   0348 B7 00C7 R        STA A TMPTRK    UPDATE TEMPS.
0642   034B F7 00C8 R        STA B TMPSCT
0643   034E 08               INX
0644   034F 08               INX
0645   0350 FF 00C5 R        STX RINTMP
0646   0353 30               TSX
0647   0354 EE 00            LDX 0,X         POINT TO FCB
0648                  ROP5  READ            GET INDEX TRACK
0649 +  0356 3F               SWI
0650 +  0357 18               FCB 24
0651   0358 E6 05            LDA B FCBSTA,X ERROR?
0652   035A 27 03            BEQ ROP5B       NO
0653                  *
0654   035C 7E 0301 R ROP5A  JMP ROP2A       YES
0655                  *
0656   035F 4D        ROP5B  TST A           TRACK=0 (END OF INDEX BLOCK)
0657   0360 27 1C            BEQ ROP6        YES
0658                  *
0659                         READ            GET INDEX SECTOR
0660 +  0362 3F               SWI
0661 +  0363 18               FCB 24
0662   0364 E6 05            LDA B FCBSTA,X ERROR?
0663   0366 26 F4            BNE ROP5A       YES
0664                  *
0665                         READ            GET INDEX POINTER
0666 +  0368 3F               SWI
0667 +  0369 18               FCB 24
0668   036A E6 05            LDA B FCBSTA,X ERROR?
0669   036C 26 EE            BNE ROP5A       YES
0670                  *
0671   036E A6 0A            LDA A FCBTRK,X NEW SECTOR?
0672   0370 E6 0B            LDA B FCBSCT,X
0673   0372 B1 00C7 R        CMP A TMPTRK
0674   0375 26 CA            BNE ROP4A       YES
0675                  *
0676   0377 F1 00C8 R        CMP B TMPSCT
0677   037A 26 C5            BNE ROP4A       YES
0678                  *
0679   037C 20 D8            BRA ROP5        NO, LOOP TIL DONE
0680                  *
0681                  * NOW POINT FILE TO FIRST DATA RECORD
0682                  *
0683   037E 6F 2E     ROP6   CLR FCBRCD,X    MAKE RCD=1
0684   0380 86 01            LDA A #1
0685   0382 A7 2F            STA A FCBRCD+1,X
0686   0384 7E 051D R        JMP POS4        POSITION FILE
0688                  * CLOSE A RANDOM-ACCESS FILE
0689                  * CALL WITH ADDRESS OF FCB IN INDEX REGISTER
0690                  *
0691                  RCLOSE PSHX           SAVE FCB ADDRESS
0692 +  0387 3F               SWI
0693 +  0388 05               FCB 5
0694                         TXAB
0695 +  0389 3F               SWI
0696 +  038A 02               FCB 2
0697                  *
0698                  * CHECK THAT FILE IS OPEN (LOOK AT FCB-CHAIN)
0699                  *
0700   038B DE 29            LDX FCBCHN
0701   038D 27 0C            BEQ RCLOS2      NO ACTIVE FCB=ERROR 13
0702                  *
0703                  RCLOS1 PSHX
0704 +  038F 3F               SWI
0705 +  0390 05               FCB 5
0706                         SUBABX          AT THIS FCB?
0707 +  0391 3F               SWI
0708 +  0392 0C               FCB 12
0709                         PULX
0710 +  0393 3F               SWI
0711 +  0394 06               FCB 6
0712   0395 27 0D            BEQ RCLOS3      YES, GOOD
0713                  *
0714   0397 EE 25            LDX FCBNFB,X    TRY NEXT FCB IF THERE IS ONE
0715   0399 26 F4            BNE RCLOS1
0716                  *
0717   039B 86 0D     RCLOS2 LDA A #13       ERROR 13
0718                  RCLERR PULX
0719 +  039D 3F               SWI
0720 +  039E 06               FCB 6
0721                         CLOSE            FORCE FILE CLOSED
0722 +  039F 3F               SWI
0723 +  03A0 15               FCB 21
0724   03A1 A7 05            STA A FCBSTA,X RETURN ERROR STATUS
0725   03A3 39               RTS
0726                  *
0727                  * NOW CHECK FOR FILETYPE=02
0728                  *
0729   03A4 30        RCLOS3 TSX
0730   03A5 EE 00            LDX 0,X         POINT AT FCB
0731   03A7 A6 1D            LDA A FCBTYP,X
0732   03A9 81 02            CMP A #2
0733   03AB 27 04            BEQ RCLOS4      YES, GOOD
0734                  *
```

```
0735   03AD 86 0E          LDA A #14      NO, ERROR 14
0736   03AF 20 EC          BRA RCLERR
0737                       *
0738   03B1 6D 06  RCLOS4  TST FCBDT1,X   READ OR WRITE?
0739   03B3 27 08          BEQ RCLOS5     IF READING, FINISH CLOSE
0740                       *
0741                       * FINISH LAST WRITE TO FILE
0742                       *
0743                       IOHDR          WRITE SECTOR
0744 +  03B5 3F            SWI
0745 +  03B6 13            FCB 19
0746   03B7 A6 05          LDA A FCBSTA,X ERROR?
0747   03B9 26 E2          BNE RCLERR     YES
0748                       *
0749   03BB 6F 06          CLR FCBDTT,X   MAKE INPUT
0750   03BD 3F     RCLOS5  PULX           RECOVER FCB ADDRESS
0751 +  03BD 3F            SWI
0752 +  03BE 06            FCB 6
0753                       CLOSE          CLOSE FILE
0754 +  03BF 3F            SWI
0755 +  03C0 15            FCB 21
0756   03C1 39             RTS
0758                       * READ A BYTE FROM A RANDOM-ACCESS FILE
0759                       * CALL WITH ADDRESS OF FCB IN INDEX REGISTER
0760                       * RETURN DATA BYTE IN "A" REGISTER
0761                       *
0762                       * WILL RETURN STATUS=15 AFTER LAST BYTE READ
0763                       *
0764   03C2 3F     RREAD   PSHX           SAVE FCB ADDRESS
0765 +  03C2 3F            SWI
0766 +  03C3 05            FCB 5
0767                       TXAB
0768 +  03C4 3F            SWI
0769 +  03C5 02            FCB 2
0770                       *
0771                       * CHECK THAT FILE IS OPEN (LOOK AT ACTIVE FCB-CHAIN)
0772                       *
0773   03C6 DE 29          LDX FCBCHN
0774   03C8 27 0C          BEQ RRED2      NO ACTIVE FCB=ERROR 13
0775                       *
0776   03CA 3F     RRED1   PSHX
0777 +  03CA 3F            SWI
0778 +  03CB 05            FCB 5
0779                       SUBABX         AT THIS FCB?
0780 +  03CC 3F            SWI
0781 +  03CD 0C            FCB 12
0782                       PULX
0783 +  03CE 3F            SWI
0784 +  03CF 06            FCB 6
0785   03D0 27 0C          BEQ RRED3      YES, GOOD
0786                       *
0787   03D2 EE 25          LDX FCBNFB,X   NO, TRY NEXT FCB IF THERE IS ONE
0788   03D4 26 F4          BNE RRED1
0789                       *
0790   03D6 86 0D  RRED2   LDA A #13      ERROR 13
0791   03D8 3F     RREDER  PULX
0792 +  03D8 3F            SWI
0793 +  03D9 06            FCB 6
0794   03DA A7 05          STA A FCBSTA,X RETURN ERROR STATUS
0795   03DC 4F             CLR A          RETURN NULL BYTE
0796   03DD 39             RTS
0797                       *
0798   03DE 30     RRED3   TSX
0799   03DF EE 00          LDX 0,X        POINT TO FCB
0800                       *
0801                       * CHECK FILETYPE=02 (RANDOM)
0802                       *
0803   03E1 A6 1D          LDA A FCBTYP,X
0804   03E3 81 02          CMP A #2
0805   03E5 27 04          BEQ RRED4      YES
0806                       *
0807   03E7 86 0E          LDA A #14      NO, ERROR 14
0808   03E9 20 ED          BRA RREDER
0809                       *
0810   03EB 6D 06  RRED4   TST FCBDTT,X   READ OR WRITE?
0811   03ED 27 08          BEQ RRED5      READ
0812                       *
0813                       * IF SWITCHING FROM WRITE, MUST FINISH LAST SECTOR
0814                       *
0815                       IOHDR          WRITE SECTOR
0816 +  03EF 3F            SWI
0817 +  03F0 13            FCB 19
0818   03F1 A6 05          LDA A FCBSTA,X ERROR?
0819   03F3 26 E3          BNE RREDER     YES
0820                       *
0821   03F5 6F 06          CLR FCBDTT,X   MAKE INPUT
0822   03F7 3F     RRED5   READ           READ DATA BYTE FROM FILE
0823 +  03F7 3F            SWI
0824 +  03F8 18            FCB 24
0825   03F9 E6 05          LDA B FCBSTA,X ERROR?
0826   03FB 27 03          BEQ RRED6      NO
0827                       *
0828   03FD 17             TBA            YES, ERROR RETURN
0829   03FE 20 D8          BRA RREDER
0830                       *
0831   0400 B7 00CA R RRED6  STA A SAVEA     SAVE DATA BYTE
0832   0403 7E 04AA R        JMP RWRIT6      UPDATE RANDOM FILE POINTERS
0834                       * WRITE A BYTE INTO A RANDOM-ACCESS FILE
0835                       * CALL WITH ADDRESS OF FCB IN INDEX REGISTER
0836                       * BYTE TO BE WRITTEN IN "A" REGISTER
0837                       *
0838                       * RETURN STATUS=15 WHEN LAST BYTE OF FILE WRITTEN
0839                       *
0840   0406 B7 00CA R RWRITE STA A SAVEA    SAVE DATA BYTE
0841                        PSHX           SAVE FCB ADDRESS
0842 +  0409 3F             SWI
0843 +  040A 05             FCB 5
0844                        TXAB
0845 +  040B 3F             SWI
0846 +  040C 02             FCB 2
0847                       *
0848                       * CHECK THAT FILE IS OPEN (LOOK AT FCB-CHAIN)
0849                       *
0850   040D DE 29          LDX FCBCHN
0851   040F 27 0C          BEQ RWRIT2     NO ACTIVE FCBS-ERROR 13
0852                       *
0853                       RWRIT1 PSHX
0854 +  0411 3F             SWI
0855 +  0412 05             FCB 5
0856                        SUBABX         AT THIS FCB?
0857 +  0413 3F             SWI
0858 +  0414 0C             FCB 12
```

```
                        PULX
 0415 3F                 SWI
 0416 06                 FCB 6
 0417 27 0B             BEQ RWRIT3     FOUND FCB?
                       *
 0419 EE 25             LDX FCBNFB,X   NO, TRY NEXT FCB IF THERE IS ONE
 041B 26 F4             BNE RWRIT1
                       *
 041D 86 0D            RWRIT2 LDA A #13      ERROR 13
                       RWTERR PULX
 041F 3F                 SWI
 0420 06                 FCB 6
 0421 A7 05             STA A FCBSTA,X RETURN ERROR STATUS
 0423 39                RTS
                       *
 0424 30               RWRIT3 TSX
 0425 EE 00             LDX 0,X        POINT TO FCB
                       *
                       * CHECK THAT FILETYPE=02 (RANDOM)
                       *
 0427 A6 1D             LDA A FCBTYP,X
 0429 81 02             CMP A #2
 042B 27 04             BEQ RWRIT4     GOOD
                       *
 042D 86 0E             LDA A #14      NO, ERROR 14
 042F 20 EE             BRA RWTERR
                       *
 0431 86 FF            RWRIT4 LDA A #$FF
 0433 A7 06             STA A FCBDTT,X        MAKE "OUTPUT"
 0435 A6 27             LDA A FCBIND,X        CHECK FOR END OF BUFFER
 0437 E6 28             LDA B FCBIND+1,X
 0439 E0 08             SUB B FCBDBA+1,X
 043B A2 07             SBC A FCBDBA,X
 043D B1 0015 R         CMP A BUFSIZ          END OF BUFFER?
 0440 26 05             BNE RWRT4A            NO
                       *
 0442 F1 0016 R         CMP B BUFSIZ+1        END OF BUFFER?
 0445 27 13             BEQ RWRT4B            YES
                       *
                       * STORE DATA BYTE INTO SECTOR BUFFER
                       *
 0447 EE 27            RWRT4A LDX FCBIND,X           POINT TO SECTOR BUFFER
 0449 B6 00CA R         LDA A SAVEA           GET DATA BYTE
 044C A7 00             STA A 0,X             STORE IT
 044E 08                INX
                        TXAB                  MOVE BUFFER POINTER
 044F 3F                 SWI
 0450 02                 FCB 2
 0451 30                TSX
 0452 EE 00             LDX 0,X
 0454 A7 27             STA A FCBIND,X        UPDATE POINTER
 0456 E7 28             STA B FCBIND+1,X
 0458 20 50             BRA RWRIT6            FINISH UP
                       *
                       * WRITE OLD SECTOR OUT AND GET NEW SECTOR
                       *
 045A 86 FF            RWRT4B LDA A #$FF             SET TO "WRITE"
 045C A7 06             STA A FCBDTT,X
                        IOHDR                 WRITE SECTOR
 045E 3F                 SWI
 045F 13                 FCB 19
 0460 A6 05             LDA A FCBSTA,X        ERROR?
 0462 26 BB             BNE RWTERR            YES
                       *
                       * BRING IN FORWARD-LINKED SECTOR AND UPDATE LINKS
                       *
 0464 A6 0A             LDA A FCBTRK,X
 0466 A1 21             CMP A FCBLTS,X        AT END OF FILE?
 0468 26 08             BNE RWRIT5            NO
                       *
 046A E6 0B             LDA B FCBSCT,X        AT END OF FILE?
 046C 26 04             BNE RWRIT5            NO
                       *
 046E 86 0F             LDA A #15             YES, ERROR 15
 0470 20 AD             BRA RWTERR
                       *
 0472 6F 06            RWRIT5 CLR FCBDTT,X           SET TO "READ"
 0474 A6 0C             LDA A FCBFWD,X
 0476 E6 0D             LDA B FCBFWD+1,X      GET FORWARD LINK T/S
 0478 A7 0A             STA A FCBTRK,X
 047A E7 0B             STA B FCBSCT,X
                        IOHDR                 READ NEW SECTOR
 047C 3F                 SWI
 047D 13                 FCB 19
 047E A6 05             LDA A FCBSTA,X        ERROR?
 0480 26 9D             BNE RWTERR            YES
                       *
 0482 EE 07             LDX FCBDBA,X          POINT TO SECTOR BUFFER
 0484 A6 00             LDA A 0,X
 0486 E6 01             LDA B 1,X             GET NEW FORWARD LINKS
 0488 30                TSX
 0489 EE 00             LDX 0,X
 048B A7 0C             STA A FCBFWD,X        UPDATE FORWARD LINKS
 048D E7 0D             STA B FCBFWD+1,X
 048F EE 07             LDX FCBDBA,X          POINT TO SECTOR BUFFER
 0491 A6 02             LDA A 2,X
 0493 E6 03             LDA B 3,X             GET NEW BACKWARD LINKS
 0495 30                TSX
 0496 EE 00             LDX 0,X
 0498 A7 0E             STA A FCBBAK,X        UPDATE BACKWARD LINKS
 049A E7 0F             STA B FCBBAK+1,X
 049C A6 07             LDA A FCBDBA,X
 049E E6 08             LDA B FCBDBA+1,X
 04A0 CB 04             ADD B #4              RE-INIT. BUFFER POINTER
 04A2 89 00             ADC A #0
 04A4 A7 27             STA A FCBIND,X
 04A6 E7 28             STA B FCBIND+1,X
 04A8 20 87             BRA RWRIT4            NOW WRITE BYTE
                       *
                       * NOW UPDATE RANDOM RECORD POINTERS
                       *
 04AA A6 30            RWRIT6 LDA A FCBPOS,X
 04AC E6 31             LDA B FCBPOS+1,X
 04AE CB 01             ADD B #1       INCREMENT POSITION POINTER
 04B0 89 00             ADC A #0
 04B2 A7 30             STA A FCBPOS,X
 04B4 E7 31             STA B FCBPOS+1,X
 04B6 A1 2C             CMP A FCBRSZ,X BEYOND RECORD LENGTH?
 04B8 22 0C             BHI RWRIT8     YES, MUST INC. RECORD NO.
                       *
 04BA 25 04             BCS RWRIT7     NO
                       *
```

```
0981   04BC E1 2D           CMP B FCBRSZ+1,X BEYOND RECORD LENGTH?
0982   04BE 22 06           BHI RWRIT8       YES, MUST INC. RECORD NO.
0983                *
0984                RWRIT7 PULX             NO, NORMAL RETURN
0985 +  04C0 3F             SWI
0986 +  04C1 06             FCB 6
0987   04C2 B6 00CA R       LDA A SAVEA
0988   04C5 39              RTS
0989                *
0990   04C6 A6 2E    RWRIT8 LDA A FCBRCD,X   UPDATE RECORD NUMBER
0991   04C8 E6 2F           LDA B FCBRCD+1,X
0992   04CA CB 01           ADD B #1
0993   04CC 89 00           ADC A #0
0994   04CE A7 2E           STA A FCBRCD,X
0995   04D0 E7 2F           STA B FCBRCD+1,X
0996   04D2 20 28           BRA POS3         POSITION FILE TO NEW RECORD
0998                * POSITION RANDOM-ACCESS FILE TO DESIRED RECORD
0999                * CALL WITH ADDRESS OF FCB IN INDEX REGISTER
1000                * DESIRED RECORD NO.  IN FCBRCD
1001                *
1002                POSITION PSHX           SAVE FCB ADDRESS
1003 +  04D4 3F             SWI
1004 +  04D5 05             FCB 5
1005                        TXAB
1006 +  04D6 3F             SWI
1007 +  04D7 02             FCB 2
1008                *
1009                * CHECK THAT FILE IS OPEN (LOOK AT FCB-CHAIN)
1010                *
1011   04D8 DE 29           LDX FCBCHN
1012   04DA 27 0C           BEQ NOCHN       NO ACTIVE FCB=ERROR 13
1013                *
1014                POS1    PSHX
1015 +  04DC 3F             SWI
1016 +  04DD 05             FCB 5
1017                        SUBABX          AT THIS FCB?
1018 +  04DE 3F             SWI
1019 +  04DF 0C             FCB 12
1020                        PULX
1021 +  04E0 3F             SWI
1022 +  04E1 06             FCB 6
1023   04E2 27 0B           BEQ POS2        YES, GOOD
1024                *
1025   04E4 EE 25           LDX FCBNFB,X    TRY NEXT FCB IF THERE IS ONE
1026   04E6 26 F4           BNE POS1
1027                *
1028   04E8 86 0D    NOCHN  LDA A #13       ERROR 13
1029                POS1A   PULX
1030 +  04EA 3F             SWI
1031 +  04EB 06             FCB 6
1032   04EC A7 05           STA A FCBSTA,X RETURN ERROR STATUS
1033   04EE 39              RTS
1034                *
1035                * NOW CHECK FOR PROPER FILETYPE=02
1036                *
1037   04EF 30      POS2    TSX
1038   04F0 EE 00           LDX 0,X         POINT TO FCB
1039   04F2 A6 1D           LDA A FCBTYP,X
1040   04F4 81 02           CMP A #2
1041   04F6 27 04           BEQ POS3        GOOD TYPE?
1042                *
1043   04F8 86 0E           LDA A #14       IF NOT, ERROR 14
1044   04FA 20 EE           BRA POS1A
1045                *
1046   04FC EE 2E   POS3    LDX FCBRCD,X    GET RECORD NUMBER
1047                *
1048                * NOW CHECK THAT RECORD NUMBER IS VALID
1049                *
1050                * 0 < FCBRCD <= FCBRNM
1051                *
1052   04FE 26 04           BNE POS3B       MUST BE >0
1053                *
1054   0500 86 0F   POS3A   LDA A #15       IF NOT, ERROR 15
1055   0502 20 E6           BRA POS1A
1056                *
1057   0504 30      POS3B   TSX
1058   0505 EE 00           LDX 0,X         POINT TO FCB
1059   0507 A6 2A           LDA A FCBRNM,X
1060   0509 E6 2B           LDA B FCBRNM+1,X
1061   050B E0 2F           SUB B FCBRCD+1,X
1062   050D A2 2E           SBC A FCBRCD,X
1063   050F 2B EF           BMI POS3A       MUST BE <=RNM
1064                *
1065                * IF FCB WAS IN "WRITE", FINISH LAST SECTOR
1066                *
1067   0511 6D 06           TST FCBDTT,X
1068   0513 27 08           BEQ POS4        IF READING, SKIP
1069                *
1070                        IOHDR           WRITE LAST SECTOR
1071 +  0515 3F             SWI
1072 +  0516 13             FCB 19
1073   0517 A6 05           LDA A FCBSTA,X ERROR?
1074   0519 26 CF           BNE POS1A       YES
1075                *
1076   051B 6F 06           CLR FCBDTT,X    MAKE INPUT
1077                *
1078                * NOW FIND PROPER FCB-TABLE ENTRY FOR RECORD
1079                *
1080                * SET "RINTMP" TO POSITION WITHIN TABLE
1081                *     "RNMTMP" TO RECORD NO.
1082                *     "A"      TO POSITION WITHIN INDEX SECTOR
1083                *     "B"      TO POSITION WITHIN INDEX BLOCK
1084                *
1085   051D 86 32   POS4    LDA A #FCBRTB
1086                        ADDAX
1087 +  051F 3F             SWI
1088 +  0520 09             FCB 9
1089   0521 FF 00C5 R       STX RINTMP      POINT TO START OF TABLE IN FCB
1090   0524 30              TSX
1091   0525 EE 00           LDX 0,X         POINT TO FCB
1092   0527 EE 2E           LDX FCBRCD,X
1093   0529 09              DEX             FIRST RECORD=0
1094   052A FF 00C1 R       STX RNMTMP      RECORD NUMBER DESIRED
1095                *
1096                * NOTE:  3 BYTES PER INDEX BLOCK
1097                *        FIRST 4 BYTES OF EACH SECTOR = LINKS
1098                *        4 BYTES OF FIRST INDEX SECTOR=RNM,RSZ
1099                *
1100   052D 86 08           LDA A #8
1101   052F C6 03           LDA B #3
1102   0531 FE 00C1 R POS4A LDX RNMTMP      AT DESIRED SECTOR?
1103   0534 27 21           BEQ POS5        IF SO, BRANCH
```

```
1104                    *
1105   0536 81 80       POS4B  CMP A #SECSIZ AT END OF AN INDEX SECTOR?
1106   0538 27 06              BEQ POS4D     IF SO, BRANCH
1107                    *
1108   053A 4C                 INC A         COUNT A BYTE
1109   053B 5A          POS4C  DEC B         COUNT A BYTE FOR INDEX BLOCK
1110   053C 27 0E              BEQ POS4E     AT END OF INDEX BLOCK?
1111                    *
1112   053E 20 F1              BRA POS4A     LOOP UNTIL DONE
1113                    *
1114                    * MOVE TABLE POINTER TO NEW SECTOR
1115                    *
1116   0540 FE 00C5 R   POS4D  LDX RINTMP
1117   0543 08                 INX
1118   0544 08                 INX
1119   0545 FF 00C5 R          STX RINTMP
1120   0548 86 04              LDA A #4      RESET "A"
1121   054A 20 EF              BRA POS4C
1122                    *
1123                    * COUNT DOWN RECORD NUMBER
1124                    *
1125   054C FE 00C1 R   POS4E  LDX RNMTMP
1126   054F 09                 DEX
1127   0550 FF 00C1 R          STX RNMTMP
1128   0553 C6 03              LDA B #3      RESET "B"
1129   0555 20 DA              BRA POS4A
1130                    *
1131                    * NOW "RINTMP" POINTS TO PROPER ENTRY IN FCB-TABLE
1132                    *     "A"      POINTS TO RECORD DATA START
1133                    *
1134   0557 B7 00C9 R   POS5   STA A TMPPNT  SAVE A
1135   055A FE 00C5 R          LDX RINTMP
1136   055D A6 00              LDA A 0,X     GET T/S FROM TABLE
1137   055F E6 01              LDA B 1,X
1138   0561 30                 TSX
1139   0562 EE 00              LDX 0,X
1140   0564 A7 0A              STA A FCBTRK,X PUT T/S INTO FCB
1141   0566 E7 0B              STA B FCBSCT,X
1142                           IOHDR         READ SECTOR OF INDEX
1143 +  0568 3F                 SWI
1144 +  0569 13                 FCB 19
1145   056A A6 05              LDA A FCBSTA,X ERROR?
1146   056C 27 03              BEQ POS5B     NO
1147                    *
1148   056E 7E 04EA R   POS5A  JMP POS1A     YES
1149                    *
1150   0571 EE 07       POS5B  LDX FCBDBA,X  POINT TO BUFFER
1151   0573 B6 00C9 R          LDA A TMPPNT
1152                           ADDAX         POINT TO RECORD DATA
1153 +  0576 3F                 SWI
1154 +  0577 09                 FCB 9
1155                           TXAB
1156 +  0578 3F                 SWI
1157 +  0579 02                 FCB 2
1158   057A 30                 TSX
1159   057B EE 00              LDX 0,X
1160   057D A7 27              STA A FCBIND,X INIT. FCBIND
1161   057F E7 28              STA B FCBIND+1,X
1162                   * NOW READ THE 3-BYTE INDEX BLOCK
1163                   *
1164                           READ          GET TRACK OF RECORD
1165 +  0581 3F                 SWI
1166 +  0582 18                 FCB 24
1167   0583 E6 05              LDA B FCBSTA,X ERROR?
1168   0585 27 04              BEQ POS5D     NO
1169                    *
1170   0587 17          POS5C  TBA           YES
1171   0588 7E 04EA R          JMP POS1A
1172                    *
1173   058B B7 00C7 R   POS5D  STA A TMPTRK
1174                           READ          GET SECTOR OF RECORD
1175 +  058E 3F                 SWI
1176 +  058F 18                 FCB 24
1177   0590 E6 05              LDA B FCBSTA,X ERROR?
1178   0592 26 F3              BNE POS5C     YES
1179                    *
1180   0594 B7 00C8 R          STA A TMPSCT
1181                           READ          GET POINTER OF RECORD
1182 +  0597 3F                 SWI
1183 +  0598 18                 FCB 24
1184   0599 E6 05              LDA B FCBSTA,X ERROR?
1185   059B 26 EA              BNE POS5C     YES
1186                    *
1187   059D B7 00C9 R          STA A TMPPNT
1188                    *
1189                    * NOW POINT FCB TO TMPTRK, TMPSCT, TMPPNT
1190                    *
1191   05A0 B6 00C7 R          LDA A TMPTRK
1192   05A3 F6 00C8 R          LDA B TMPSCT
1193   05A6 A7 0A              STA A FCBTRK,X      SET TRACK
1194   05A8 E7 0B              STA B FCBSCT,X      SET SECTOR
1195   05AA 6F 06              CLR FCBDTT,X        MAKE "INPUT"
1196                           IOHDR               READ IN DATA SECTOR
1197 +  05AC 3F                 SWI
1198 +  05AD 13                 FCB 19
1199   05AE A6 05              LDA A FCBSTA,X      ERROR?
1200   05B0 26 BC              BNE POS5A           YES
1201                    *
1202   05B2 EE 07              LDX FCBDBA,X
1203   05B4 A6 00              LDA A 0,X           GET FORWARD LINKS
1204   05B6 E6 01              LDA B 1,X
1205   05B8 30                 TSX
1206   05B9 EE 00              LDX 0,X
1207   05BB A7 0C              STA A FCBFWD,X      UPDATE LINKAGE
1208   05BD E7 0D              STA B FCBFWD+1,X
1209   05BF EE 07              LDX FCBDBA,X        POINT TO SECTOR BUFFER
1210   05C1 A6 02              LDA A 2,X
1211   05C3 E6 03              LDA B 3,X           GET BACKWARD LINKS
1212   05C5 30                 TSX
1213   05C6 EE 00              LDX 0,X
1214   05C8 A7 0E              STA A FCBBAK,X      UPDATE LINKAGE
1215   05CA E7 0F              STA B FCBBAK+1,X
1216   05CC EE 07              LDX FCBDBA,X
1217   05CE B6 00C9 R          LDA A TMPPNT
1218                           ADDAX
1219 +  05D1 3F                 SWI
1220 +  05D2 09                 FCB 9
1221                           TXAB
1222 +  05D3 3F                 SWI
1223 +  05D4 02                 FCB 2
1224                           PULX                RECOVER FCB ADDRESS
1225 +  05D5 3F                 SWI
```

```
1226 +  05D6 06                FCB 6
1227    05D7 A7 27             STA A FCBIND,X       SET BUFFER POINTER
1228    05D9 E7 28             STA B FCBIND+1,X
1229    05DB 6F 30             CLR FCBPOS,X         INIT. RECORD POINTER
1230    05DD 86 01             LDA A #1
1231    05DF A7 31             STA A FCBPOS+1,X
1232    05E1 B6 00CA R         LDA A SAVEA          GOT HERE FROM "RREAD","RWRITE"
1233    05E4 39                RTS
1234                  *
1236                  * EXPAND A RANDOM-ACCESS FILE
1237                  * CALLED WITH ADDRESS OF FCB IN INDEX REGISTER
1238                  * FILE MUST ALREADY BE OPEN
1239                  * NUMBER OF RECORDS TO ADD IN FCBRCD
1240                  * RECORDS WILL HAVE SAME SIZE AS ORIGINALS
1241                  *
1242                  EXPAND PSHX          SAVE FCB ADDRESS
1243 +  05E5 3F                SWI
1244 +  05E6 05                FCB 5
1245                           TXAB
1246 +  05E7 3F                SWI
1247 +  05E8 02                FCB 2
1248                  * CHECK THAT FILE IS OPEN (LOOK AT FCB-CHAIN)
1249    05E9 DE 29             LDX FCBCHN
1250    05EB 27 08             BEQ EXP2       NO ACTIVE FCBS=ERROR 13
1251                  *
1252                  EXP1   PSHX
1253 +  05ED 3F                SWI
1254 +  05EE 05                FCB 5
1255                           SUBABX         AT THIS FCB?
1256 +  05EF 3F                SWI
1257 +  05F0 0C                FCB 12
1258                           PULX
1259 +  05F1 3F                SWI
1260 +  05F2 06                FCB 6
1261    05F3 27 0C             BEQ EXP3       YES
1262                  *
1263    05F5 86 0D     EXP2    LDA A #13      NO, ERROR 13
1264    05F7 CE 0017 R EXPERR  LDX #RNDFCB
1265                           CLOSE          FORCE FILES CLOSED
1266 +  05FA 3F                SWI
1267 +  05FB 15                FCB 21
1268                           PULX
1269 +  05FC 3F                SWI
1270 +  05FD 06                FCB 6
1271    05FE A7 05             STA A FCBSTA,X RETURN ERROR STATUS
1272    0600 39                RTS
1273                  *
1274    0601 30        EXP3    TSX
1275    0602 EE 00             LDX 0,X
1276                  * CHECK THAT FILE IS RANDOM-ACCESS (TYPE=02)
1277    0604 A6 1D             LDA A FCBTYP,X
1278    0606 81 02             CMP A #2       RANDOM?
1279    0608 27 04             BEQ EXP4       YES, GOOD
1280                  *
1281    060A 86 0E             LDA A #14      NO, ERROR 14
1282    060C 20 E9             BRA EXPERR
1283                  *
1284    060E EE 2E     EXP4    LDX FCBRCD,X   GET NO. OF NEW RECORDS
1285    0610 26 04             BNE EXP4B      NONZERO
1286                  *
1287    0612 86 0C     EXP4A   LDA A #12      ZERO-ERROR 12
1288    0614 20 E1             BRA EXPERR
1289                  *
1290    0616 30        EXP4B   TSX
1291    0617 EE 00             LDX 0,X
1292                  * CHECK THAT NEW RECNUM= OLD RECNUM+FCBRCD < MXRNUM
1293    0619 A6 2A             LDA A FCBRNM,X
1294    061B E6 2B             LDA B FCBRNM+1,X
1295    061D EB 2F             ADD B FCBRCD+1,X
1296    061F A9 2E             ADC A FCBRCD,X
1297    0621 CE 09B0           LDX #MXRNUM          <MXRNUM?
1298                           SUBABX
1299 +  0624 3F                 SWI
1300 +  0625 0C                 FCB 12
1301    0626 2B EA             BMI EXP4A            NO, ERROR 12
1302                  *
1303    0628 30        EXP5    TSX
1304    0629 EE 00             LDX 0,X
1305    062B A7 2A             STA A FCBRNM,X       UPDATE RNUM
1306    062D E7 2B             STA B FCBRNM+1,X
1307    062F A6 1F             LDA A FCBFTS,X
1308    0631 E6 20             LDA B FCBFTS+1,X     POINT FILE TO FIRST T/S
1309    0633 A7 0A             STA A FCBTRK,X
1310    0635 E7 0B             STA B FCBSCT,X
1311    0637 6F 06             CLR FCBDTT,X         MAKE 'INPUT'
1312    0639 6F 29             CLR FCBSCF,X         NO SPACE COMPRESSION
1313    063B A6 21             LDA A FCBLTS,X
1314    063D E6 22             LDA B FCBLTS+1,X
1315    063F B7 00C7 R         STA A TMPTRK         SAVE LAST T/S
1316    0642 F7 00C8 R         STA B TMPSCT
1317                           IOHDR                READ FIRST T/S OF FILE
1318 +  0645 3F                 SWI
1319 +  0646 13                 FCB 19
1320    0647 A6 05             LDA A FCBSTA,X       ERROR?
1321    0649 27 03             BEQ EXP6             NO
1322                  *
1323    064B 7E 05F7 R EXP5A   JMP EXPERR           YES
1324                  *
1325    064E A6 2A     EXP6    LDA A FCBRNM,X       GET NEW RNUM
1326    0650 E6 2B             LDA B FCBRNM+1,X
1327    0652 EE 07             LDX FCBDBA,X         POINT TO SECTOR BUFFER
1328    0654 A7 04             STA A 4,X            PUT NEW RNUM IN BUFFER
1329    0656 E7 05             STA B 5,X
1330    0658 A6 00             LDA A 0,X            GET FORWARD LINKS
1331    065A E6 01             LDA B 1,X
1332    065C 30                TSX
1333    065D EE 00             LDX 0,X
1334    065F A7 0C             STA A FCBFWD,X       UPDATE FORWARD LINKAGE
1335    0661 E7 0D             STA B FCBFWD+1,X
1336    0663 EE 07             LDX FCBDBA,X         POINT TO SECTOR BUFFER
1337    0665 A6 02             LDA A 2,X            GET BACKWARD LINKS
1338    0667 E6 03             LDA B 3,X
1339    0669 30                TSX
1340    066A EE 00             LDX 0,X
1341    066C A7 0E             STA A FCBBAK,X       UPDATE BACKWARD LINKAGE
1342    066E E7 0F             STA B FCBBAK+1,X
1343    0670 EE 07             LDX FCBDBA,X
1344    0672 86 08             LDA A #8             SKIP FIRST 8 BYTES OF BUFFER
1345                           ADDAX
1346 +  0674 3F                 SWI
1347 +  0675 09                 FCB 9
1348                           TXAB                 NEW BUFFER POINTER
```

```
1349 +  0676 3F                SWI
1350 +  0677 02                FCB 2
1351    0678 30                TSX
1352    0679 EE 00             LDX 0,X
1353    067B A7 27             STA A FCBIND,X          UPDATE BUFFER POINTER
1354    067D E7 28             STA B FCBIND+1,X
1355                   EXP7    READ                    READ INDEX FROM FILE
1356 +  067F 3F                 SWI
1357 +  0680 18                 FCB 24
1358    0681 E6 05             LDA B FCBSTA,X          ERROR?
1359    0683 27 04             BEQ EXP7B               NO
1360                   *
1361    0685 17        EXP7A   TBA                     YES
1362    0686 7E 05F7 R         JMP EXPERR
1363                   *
1364    0689 4D        EXP7B   TST A                   INDEX TRACK=0? (END OF INDEX)
1365    068A 27 0E             BEQ EXP8                YES
1366                   *
1367                           READ
1368 +  068C 3F                 SWI
1369 +  068D 18                 FCB 24
1370    068E E6 05             LDA B FCBSTA,X          ERROR?
1371    0690 26 F3             BNE EXP7A               YES
1372                   *
1373                           READ
1374 +  0692 3F                 SWI
1375 +  0693 18                 FCB 24
1376    0694 E6 05             LDA B FCBSTA,X          ERROR?
1377    0696 26 ED             BNE EXP7A               YES
1378                   *
1379    0698 20 E5             BRA EXP7                LOOP UNTIL END OF INDEX FOUND
1380                   *
1381                   * FOUND END OF OLD INDEX, NOW START ADDING NEW STUFF
1382                   *
1383    069A 63 06     EXP8    COM FCBDTT,X            MAKE 'OUTPUT'
1384    069C A6 27             LDA A FCBIND,X
1385    069E E6 28             LDA B FCBIND+1,X
1386    06A0 C0 01             SUB B #1                BACK UP ONE BYTE IN BUFFER
1387    06A2 82 00             SBC A #0
1388    06A4 A7 27             STA A FCBIND,X
1389    06A6 E7 28             STA B FCBIND+1,X
1390    06A8 A6 0C             LDA A FCBFWD,X          GET FORWARD LINK
1391    06AA E6 0D             LDA B FCBFWD+1,X
1392    06AC B7 00C5 R         STA A RINTMP            SAVE T/S OF OLD DATA START
1393    06AF F7 00C6 R         STA B RINTMP+1
1394                   *
1395                   * RINTMP IS FIRST SECTOR OF OLD DATA RECORDS
1396                   * NOW INITIALIZE DATA FCB AND MAKE NEW RECORDS
1397                   *
1398    06B2 E6 09             LDA B FCBDRV,X          GET DRIVE NO. OF FILE
1399    06B4 CE 0017 R         LDX #RNDFCB
1400    06B7 6F 05             CLR FCBSTA,X            INIT. DATA STATUS
1401    06B9 6F 29             CLR FCBSCF,X            NO SPACE COMPRESSION
1402    06BB 86 FF             LDA A #$FF
1403    06BD A7 06             STA A FCBDTT,X          "OUTPUT"
1404    06BF 86 20             LDA A #$20
1405    06C1 A7 10             STA A FCBNAM,X          NAME=BLANK
1406    06C3 E7 09             STA B FCBDRV,X          INIT. DRIVE NO.
1407                           OPEN                    OPEN DATA FILE
1408 +  06C5 3F                 SWI
1409 +  06C6 14                 FCB 20
1410    06C7 A6 05             LDA A FCBSTA,X          ERROR?
1411    06C9 27 03             BEQ EXP8B               NO
1412                   *
1413    06CB 7E 05F7 R EXP8A   JMP EXPERR              YES
1414                   *
1415    06CE A6 1F     EXP8B   LDA A FCBFTS,X          GET FIRST T/S OF DATA
1416    06D0 E6 20             LDA B FCBFTS+1,X
1417    06D2 30                TSX
1418    06D3 EE 00             LDX 0,X
1419                           WRITE                   WRITE INDEX TRACK
1420 +  06D5 3F                 SWI
1421 +  06D6 19                 FCB 25
1422    06D7 A6 05             LDA A FCBSTA,X          ERROR?
1423    06D9 26 F0             BNE EXP8A               YES
1424                   *
1425    06DB 17                TBA
1426                           WRITE                   WRITE INDEX SECTOR
1427 +  06DC 3F                 SWI
1428 +  06DD 19                 FCB 25
1429    06DE A6 05             LDA A FCBSTA,X          ERROR?
1430    06E0 26 E9             BNE EXP8A               YES
1431                   *
1432    06E2 86 04             LDA A #4                FIRST INDEX POINTER=4
1433                           WRITE
1434 +  06E4 3F                 SWI
1435 +  06E5 19                 FCB 25
1436    06E6 A6 05             LDA A FCBSTA,X          ERROR?
1437    06E8 26 E1             BNE EXP8A               YES
1438                   *
1439    06EA A6 2E             LDA A FCBRCD,X
1440    06EC E6 2F             LDA B FCBRCD+1,X
1441    06EE B7 00C1 R         STA A RNMTMP            INIT. TEMP. RECNUM
1442    06F1 F7 00C2 R         STA B RNMTMP+1
1443                   *
1444                   * LOOP THROUGH HERE FOR EACH NEW DATA RECORD
1445                   *
1446    06F4 A6 2C     EXP9    LDA A FCBRSZ,X
1447    06F6 E6 2D             LDA B FCBRSZ+1,X
1448    06F8 B7 00C3 R         STA A RSZTMP            INIT. TEMP. RECSIZ
1449    06FB F7 00C4 R         STA B RSZTMP+1
1450    06FE CE 0017 R EXP9A   LDX #RNDFCB             POINT TO DATA FCB
1451    0701 4F                CLR A
1452                           WRITE                   WRITE NULL TO DATA
1453 +  0702 3F                 SWI
1454 +  0703 19                 FCB 25
1455    0704 A6 05             LDA A FCBSTA,X          ERROR?
1456    0706 26 C3             BNE EXP8A               YES
1457                   *
1458    0708 FE 00C3 R         LDX RSZTMP
1459    070B 09                DEX                     COUNT DOWN RECSIZ
1460    070C FF 00C3 R         STX RSZTMP
1461    070F 26 ED             BNE EXP9A               LOOP UNTIL RECSIZ=0
1462                   *
1463    0711 FE 00C1 R         LDX RNMTMP
1464    0714 09                DEX                     COUNT DOWN RECNUM
1465    0715 FF 00C1 R         STX RNMTMP
1466    0718 27 29             BEQ EXP10               DONE?
1467                   *
1468                   * OUTPUT INDEX BLOCK FOR NEW RECORD
1469                   *
1470    071A CE 0017 R         LDX #RNDFCB
```

```
1471   071D A6 0A           LDA A FCBTRK,X      GET T/S OF DATA RECORD
1472   071F E6 0B           LDA B FCBSCT,X
1473   0721 30              TSX
1474   0722 EE 00           LDX 0,X
1475                        WRITE               WRITE INDEX TRACK
1476 +  0724 3F             SWI
1477 +  0725 19             FCB 25
1478   0726 A6 05           LDA A FCBSTA,X      ERROR?
1479   0728 26 A1           BNE EXP8A           YES
1480                *
1481   072A 17              TBA
1482                        WRITE               WRITE INDEX SECTOR
1483 +  072B 3F             SWI
1484 +  072C 19             FCB 25
1485   072D A6 05           LDA A FCBSTA,X      ERROR?
1486   072F 26 9A           BNE EXP8A           YES
1487                *
1488   0731 CE 0017 R       LDX #RNDFCB
1489   0734 A6 28           LDA A FCBIND+1,X
1490   0736 A0 08           SUB A FCBDBA+1,X    FIND POINTER
1491   0738 30              TSX
1492   0739 EE 00           LDX 0,X
1493                        WRITE               WRITE INDEX POINTER
1494 +  073B 3F             SWI
1495 +  073C 19             FCB 25
1496   073D A6 05           LDA A FCBSTA,X      ERROR?
1497   073F 26 8A           BNE EXP8A           YES
1498                *
1499   0741 20 B1           BRA EXP9            LOOP UNTIL ALL RECORDS DONE
1500                *
1501                * NOW PATCH LINKAGES FROM NEW STUFF TO OLD
1502                * FIRST, CLOSE DATA FILE WITH NO DIRECTORY UPDATE
1503                *
1504   0743 CE 0017 R EXP10  LDX #RNDFCB
1505                        TXAB                DATA FCB ADDRESS IN (A,B)
1506 +  0746 3F             SWI
1507 +  0747 02             FCB 2
1508   0748 DE 29           LDX FCBCHN          GET HEAD OF FCB-CHAIN
1509                * DELETE FCB FROM ACTIVE FCB-CHAIN
1510                        PSHX
1511 +  074A 3F             SWI
1512 +  074B 05             FCB 5
1513                        SUBABX              AT THIS FCB?
1514 +  074C 3F             SWI
1515 +  074D 0C             FCB 12
1516                        PULX
1517 +  074E 3F             SWI
1518 +  074F 06             FCB 6
1519   0750 26 0A           BNE EXP10A          NO
1520                *
1521   0752 A6 25           LDA A FCBNFB,X
1522   0754 E6 26           LDA B FCBNFB+1,X
1523   0756 97 29           STA A FCBCHN        MAKE NEW CHAIN HEAD
1524   0758 D7 2A           STA B FCBCHN+1
1525   075A 20 1C           BRA EXP10C
1526                *
1527   075C A1 25     EXP10A CMP A FCBNFB,X     AT DESIRED FCB?
1528   075E 26 14           BNE EXP10B          NO
1529                *
1530   0760 E1 26           CMP B FCBNFB+1,X    AT DESIRED FCB?
1531   0762 26 10           BNE EXP10B          NO
1532                *
1533                * FIX FCB-CHAIN TO GO AROUND THIS FCB
1534                *
1535                        PSHX
1536 +  0764 3F             SWI
1537 +  0765 05             FCB 5
1538                        TABX
1539 +  0766 3F             SWI
1540 +  0767 03             FCB 3
1541   0768 A6 25           LDA A FCBNFB,X
1542   076A E6 26           LDA B FCBNFB+1,X
1543                        PULX
1544 +  076C 3F             SWI
1545 +  076D 06             FCB 6
1546   076E A7 25           STA A FCBNFB,X
1547   0770 E7 26           STA B FCBNFB+1,X
1548   0772 20 04           BRA EXP10C
1549                *
1550   0774 EE 25     EXP10B LDX FCBNFB,X       GET NEXT FCB IN CHAIN
1551   0776 20 E4           BRA EXP10A          LOOP
1552                *
1553   0778 CE 0017 R EXP10C LDX #RNDFCB
1554                *
1555                * WRITE OUT LAST DATA SECTOR TO FILE
1556                *
1557   077B 6D 0A           TST FCBTRK,X        AT END OF DISK?
1558   077D 27 04           BEQ EXP10D          YES
1559                *
1560   077F 6D 0B           TST FCBSCT,X        AT END OF DISK?
1561   0781 26 0A           BNE EXP10E          NO
1562                *
1563   0783 A6 0E     EXP10D LDA A FCBBAK,X     FIX-UP FOR END OF DISK
1564   0785 E6 0F           LDA B FCBBAK+1,X
1565   0787 A7 0A           STA A FCBTRK,X
1566   0789 E7 0B           STA B FCBSCT,X
1567   078B 20 15           BRA EXP10H
1568                *
1569                EXP10E  IOHDR               WRITE LAST SECTOR
1570 +  078D 3F             SWI
1571 +  078E 13             FCB 19
1572   078F A6 05           LDA A FCBSTA,X      ERROR?
1573   0791 27 03           BEQ EXP10G          NO
1574                *
1575   0793 7E 05F7 R EXP10F JMP EXPERR          YES
1576                *
1577   0796 A6 23     EXP10G LDA A FCBNMS,X     ONE MORE SECTOR IN COUNT
1578   0798 E6 24           LDA B FCBNMS+1,X
1579   079A CB 01           ADD B #1
1580   079C 89 00           ADC A #0
1581   079E A7 23           STA A FCBNMS,X
1582   07A0 E7 24           STA B FCBNMS+1,X
1583   07A2 A6 0A     EXP10H LDA A FCBTRK,X     GET LAST T/S
1584   07A4 E6 0B           LDA B FCBSCT,X
1585   07A6 A7 21           STA A FCBLTS,X      UPDATE LT,LS
1586   07A8 E7 22           STA B FCBLTS+1,X
1587   07AA 6F 06           CLR FCBDTT,X        MAKE 'INPUT'
1588   07AC 86 00           LDA A #0            TRACK=0
1589   07AE C6 03           LDA B #FRESEC       FREE-SPACE SECTOR
1590   07B0 A7 0A           STA A FCBTRK,X
1591   07B2 E7 0B           STA B FCBSCT,X
1592                        IOHDR               READ FREE-SPACE SECTOR
```

```
1593 +  07B4 3F              SWI
1594 +  07B5 13              FCB 19
1595    07B6 A6 05           LDA A FCBSTA,X       ERROR?
1596    07B8 26 D9           BNE EXP10F           YES
1597                 *
1598    07BA 63 06           COM FCBDTT,X         MAKE 'OUTPUT'
1599    07BC A6 09           LDA A FCBDRV,X       GET DRIVE NO.
1600    07BE 84 03           AND A #$03           LIMIT RANGE (0-3)
1601    07C0 48              ASL A                2 BYTES/TABLE ENTRY
1602    07C1 CE 002B         LDX #FRETAB
1603                         ADDAX                ACCESS FREE-SPACE TABLE
1604 +  07C4 3F              SWI
1605 +  07C5 09              FCB 9
1606    07C6 A6 00           LDA A 0,X
1607    07C8 E6 01           LDA B 1,X            GET FREE T/S
1608    07CA CE 0041 R       LDX #RNDBUF          POINT TO DATA SECTOR BUFFER
1609    07CD A7 7E           STA A SECSIZ-2,X     PUT NEW T/S INTO BUFFER
1610    07CF E7 7F           STA B SECSIZ-1,X
1611    07D1 CE 0017 R       LDX #RNDFCB
1612                         IOHDR                WRITE UPDATED FREE-SPACE SECTOR
1613 +  07D4 3F              SWI
1614 +  07D5 13              FCB 19
1615    07D6 A6 05           LDA A FCBSTA,X       ERROR?
1616    07D8 26 B9           BNE EXP10F           YES
1617                 *
1618                 * WRITE LAST INDEX BLOCK=0, 0, 0
1619                 *
1620    07DA 30              TSX
1621    07DB EE 00           LDX 0,X
1622    07DD 4F              CLR A
1623                         WRITE                WRITE INDEX TRACK=0
1624 +  07DE 3F              SWI
1625 +  07DF 19              FCB 25
1626    07E0 A6 05           LDA A FCBSTA,X       ERROR?
1627    07E2 26 AF           BNE EXP10F           YES
1628                 *
1629                         WRITE                WRITE INDEX SECTOR=0
1630 +  07E4 3F              SWI
1631 +  07E5 19              FCB 25
1632    07E6 A6 05           LDA A FCBSTA,X       ERROR?
1633    07E8 26 A9           BNE EXP10F           YES
1634                 *
1635                         WRITE                WRITE INDEX POINTER=0
1636 +  07EA 3F              SWI
1637 +  07EB 19              FCB 25
1638    07EC A6 05           LDA A FCBSTA,X       ERROR?
1639    07EE 26 A3           BNE EXP10F           YES
1640                 *
1641                 * RE-LINK INDEX TO DATA
1642                 *
1643    07F0 CE 0017 R       LDX #RNDFCB
1644    07F3 A6 1F           LDA A FCBFTS,X       GET FIRST T/S OF DATA
1645    07F5 E6 20           LDA B FCBFTS+1,X
1646    07F7 30              TSX
1647    07F8 EE 00           LDX 0,X              POINT TO INDEX FCB
1648    07FA EE 07           LDX FCBDBA,X         POINT TO SECTOR BUFFER
1649    07FC A7 00           STA A 0,X            UPDATE FORWARD LINKS
1650    07FE E7 01           STA B 1,X
1651    0800 30              TSX
1652    0801 EE 00           LDX 0,X
1653                         IOHDR                WRITE LAST INDEX SECTOR
1654 +  0803 3F              SWI
1655 +  0804 13              FCB 19
1656    0805 A6 05           LDA A FCBSTA,X       ERROR?
1657    0807 26 16           BNE EXP11            YES
1658                 *
1659    0809 A6 0A           LDA A FCBTRK,X       GET T/S OF INDEX SECTOR
1660    080B E6 0B           LDA B FCBSCT,X
1661    080D 36              PSH A
1662    080E A6 1F           LDA A FCBFTS,X       REWIND DATA FCB
1663    0810 A7 0A           STA A FCBTRK,X
1664    0812 A6 20           LDA A FCBFTS+1,X
1665    0814 A7 0B           STA A FCBSCT,X
1666    0816 6F 06           CLR FCBDTT,X         MAKE 'INPUT'
1667                         IOHDR                READ FIRST DATA SECTOR
1668 +  0818 3F              SWI
1669 +  0819 13              FCB 19
1670    081A A6 05           LDA A FCBSTA,X       ERROR?
1671    081C 27 04           BEQ EXP11A           NO
1672                 *
1673    081E 31              INS                  CLEAN STACK
1674    081F 7E 05F7 R EXP11 JMP EXPERR
1675                 *
1676    0822 32              EXP11A PUL A
1677    0823 63 06           COM FCBDTT,X         MAKE 'OUTPUT'
1678    0825 EE 07           LDX FCBDBA,X         POINT TO SECTOR BUFFER
1679    0827 A7 02           STA A 2,X            UPDATE BACKWARD LINKS
1680    0829 E7 03           STA B 3,X
1681    082B CE 0017 R       LDX #RNDFCB
1682                         IOHDR                WRITE SECTOR
1683 +  082E 3F              SWI
1684 +  082F 13              FCB 19
1685    0830 A6 05           LDA A FCBSTA,X       ERROR?
1686    0832 26 EB           BNE EXP11            YES
1687                 *
1688                 * LINK NEW DATA SECTORS TO OLD DATA
1689                 *
1690    0834 A6 21           LDA A FCBLTS,X       GET LAST T/S OF DATA
1691    0836 E6 22           LDA B FCBLTS+1,X
1692    0838 A7 0A           STA A FCBTRK,X
1693    083A E7 0B           STA B FCBSCT,X
1694    083C 6F 06           CLR FCBDTT,X         MAKE 'INPUT'
1695                         IOHDR                READ SECTOR
1696 +  083E 3F              SWI
1697 +  083F 13              FCB 19
1698    0840 A6 05           LDA A FCBSTA,X       ERROR?
1699    0842 26 DB           BNE EXP11            YES
1700                 *
1701    0844 B6 00C5 R       LDA A RINTMP         GET FIRST T/S OF OLD DATA
1702    0847 F6 00C6 R       LDA B RINTMP+1
1703                         PSHX
1704 +  084A 3F              SWI
1705 +  084B 05              FCB 5
1706    084C EE 07           LDX FCBDBA,X         POINT TO SECTOR BUFFER
1707    084E A7 00           STA A 0,X            UPDATE FORWARD LINKS
1708    0850 E7 01           STA B 1,X
1709                         PULX
1710 +  0852 3F              SWI
1711 +  0853 06              FCB 6
1712    0854 63 06           COM FCBDTT,X         MAKE 'OUTPUT'
1713                         IOHDR                WRITE SECTOR
1714 +  0856 3F              SWI
```

```
1715 +  0857 13                FCB 19
1716    0858 A6 05             LDA A FCBSTA,X       ERROR?
1717    085A 26 C3             BNE EXP11            YES
1718                   *
1719    085C 6F 06             CLR FCBDTT,X         MAKE 'INPUT'
1720    085E B6 00C5 R         LDA A RINTMP         GET FIRST T/S OF OLD DATA
1721    0861 F6 00C6 R         LDA B RINTMP+1
1722    0864 A7 0A             STA A FCBTRK,X
1723    0866 E7 0B             STA B FCBSCT,X
1724                           IOHDR                READ SECTOR
1725 +  0868 3F                SWI
1726 +  0869 13                FCB 19
1727    086A A6 05             LDA A FCBSTA,X       ERROR?
1728    086C 26 B1             BNE EXP11            YES
1729                   *
1730    086E A6 21             LDA A FCBLTS,X       GET LAST T/S OF NEW DATA
1731    0870 E6 22             LDA B FCBLTS+1,X
1732                           PSHX
1733 +  0872 3F                SWI
1734 +  0873 05                FCB 5
1735    0874 EE 07             LDX FCBDBA,X         POINT TO SECTOR BUFFER
1736    0876 A7 02             STA A 2,X            UPDATE BACKWARD LINKS
1737    0878 E7 03             STA B 3,X
1738                           PULX
1739 +  087A 3F                SWI
1740 +  087B 06                FCB 6
1741    087C 63 06             COM FCBDTT,X         MAKE 'OUTPUT'
1742                           IOHDR                WRITE SECTOR
1743 +  087E 3F                SWI
1744 +  087F 13                FCB 19
1745    0880 A6 05             LDA A FCBSTA,X       ERROR?
1746    0882 26 9B             BNE EXP11            YES
1747                   *
1748    0884 A6 23             LDA A FCBNMS,X       GET NO. OF SECTORS
1749    0886 E6 24             LDA B FCBNMS+1,X
1750    0888 30                TSX
1751    0889 EE 00             LDX 0,X
1752    088B EB 24             ADD B FCBNMS+1,X     TOTAL BOTH FILES
1753    088D A9 23             ADC A FCBNMS,X
1754    088F C0 01             SUB B #1             DECREMENT (CLOSE WILL ADD BACK)
1755    0891 82 00             SBC A #0
1756    0893 A7 23             STA A FCBNMS,X
1757    0895 E7 24             STA B FCBNMS+1,X
1758    0897 B6 00C7 R         LDA A TMPTRK         POINT FILE TO END
1759    089A F6 00C8 R         LDA B TMPSCT
1760    089D 6F 06             CLR FCBDTT,X         MAKE 'INPUT'
1761    089F A7 0A             STA A FCBTRK,X
1762    08A1 E7 0B             STA B FCBSCT,X
1763                           IOHDR                READ LAST SECTOR
1764 +  08A3 3F                SWI
1765 +  08A4 13                FCB 19
1766    08A5 A6 05             LDA A FCBSTA,X       ERROR?
1767    08A7 27 03             BEQ EXP12            NO
1768                   *
1769    08A9 7E 05F7 R         JMP EXPERR           YES
1770                   *
1771    08AC 63 06     EXP12   COM FCBDTT,X         MAKE 'OUTPUT'
1772                           CLOSE                CLOSE FILE
1773 +  08AE 3F                SWI
1774 +  08AF 15                FCB 21
1775                           PULX
1776 +  08B0 3F                SWI
1777 +  08B1 06                FCB 6
1778    08B2 39                RTS
1779                   *
1780                           END
```

```
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
BMEM   0033
BS     0039
BUFSIZ 0015 R
CHAIN  243A M
CLASS  0026
CLOSE  2369 M
CMEM   0037
CMPC   231B M
CMWC   2572 M
CR2    00EB R
CR3    00F8 R
CR4    00FC R
CR5    0109 R
CR5A   014A R
CR6    014D R
CR7    0176 R
CR7A   0180 R
CR7B   01AC R
CR7C   01AF R
CR8    01C9 R
CR8A   01E2 R
CR8B   01FA R
CR8C   01FE R
CR8D   0209 R
CR8E   0213 R
CR8F   0225 R
CR9    025D R
CR9A   029F R
CR9B   02A2 R
CREATE 00CB RN
CRERR  00E1 R
CUCHAR 0023
DELETE 2420 M
DESCRA 0020
DESCRC 0022
DIV16  2524 M
DL     003A
DP     003B
DPCNT  003C
DX     0040
EJ     0041
EMEM   0035
ES     0043
EXP1   05ED R
EXP10  0743 R
EXP10A 075C R
EXP10B 0774 R
EXP10C 0778 R
EXP10D 0783 R
EXP10E 078D R
EXP10F 0793 R
EXP10G 0796 R
EXP10H 07A2 R
EXP11  081F R
EXP11A 0822 R
EXP12  08AC R
EXP2   05F5 R
EXP3   0601 R
EXP4   060E R
EXP4A  0612 R
EXP4B  0616 R
EXP5   0628 R
EXP5A  064B R
EXP6   064E R
EXP7   067F R
EXP7A  0685 R
EXP7B  0689 R
EXP8   069A R
EXP8A  06CB R
EXP8B  06CE R
EXP9   06F4 R
EXP9A  06FE R
EXPAND 05E5 RN
EXPERR 05F7 R
FCBACS 001E
FCBBAK 000E
FCBCHN 0029
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBDTT 0006
FCBEQT 0000
FCBFTS 001F
FCBFWD 000C
FCBGDT 0002
FCBIND 0027
FCBLTS 0021
FCBNAM 0010
FCBNFB 0025
FCBNMS 0023
FCBPOS 0030
FCBRCD 002E
FCBRNM 002A
FCBRSZ 002C
FCBRTB 0032
FCBRTE 00AA
FCBSCF 0029
FCBSCT 000B
FCBSTA 0005
FCBTRK 000A
FCBTYP 001D
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
FRESEC 0003
FRETAB 002B
GETDR  23EC M
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
IOHDR  2335 M
LDP    0044
LDPCNT 0045
LOADB  246E M
LWD    0046
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
MXRNUM 09B0
NL     003E
NOCHN  04E8 R
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
POS1   04DC R
POS1A  04EA R
POS2   04EF R
POS3   04FC R
POS3A  0500 R
POS3B  0504 R
POS4   051D R
POS4A  0531 R
POS4B  0536 R
POS4C  053B R
POS4D  0540 R
POS4E  054C R
POS5   0557 R
POS5A  056E R
POS5B  0571 R
POS5C  0587 R
POS5D  058B R
POSITI 04D4 RN
PRTERR 2454 M
PRTMSG 250A M
PS     0042
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
RANDOM 0000 RN
RC     0025
RCBDEF 258C M
RCLERR 039D R
RCLOS1 038F R
RCLOS2 039B R
RCLOS3 03A4 R
RCLOS4 03B1 R
RCLOS5 03BD R
RCLOSE 0387 RN
READ   23B8 M
REWIND 2384 M
RINTMP 00C5 R
RNDBUF 0041 R
RNDFCB 0017 R
RNDVEC 0000 R
RNMTMP 00C1 R
ROP2   02F9 R
ROP2A  0301 R
ROP3   0305 R
ROP3A  030B R
ROP3B  030E R
ROP4   032A R
ROP4A  0341 R
ROP5   0356 R
ROP5A  035C R
ROP5B  035F R
ROP6   037E R
ROPEN  02E6 RN
ROPERR 02F4 R
RREAD  03C2 RN
RRED1  03CA R
RRED2  03D6 R
RRED3  03DE R
RRED4  03EB R
RRED5  03F7 R
RRED6  0400 R
RREDER 03D8 R
RSZTMP 00C3 R
RWRIT1 0411 R
RWRIT2 041D R
RWRIT3 0424 R
RWRIT4 0431 R
RWRIT5 0472 R
RWRIT6 04AA R
RWRIT7 04C0 R
RWRIT8 04C6 R
RWRITE 0406 RN
RWRT4A 0447 R
RWRT4B 045A R
RWTERR 041F R
SAVEA  00CA R
SECSIZ 0080
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
TABX   219C M
TB     003F
TMPPNT 00C9 R
TMPSCT 00C8 R
TMPTRK 00C7 R
TRKSIZ 001A
TXAB   2183 M
VALUE  0027
WD     003D
WRITE  23D2 M
XABX   21B5 M
```

```
S0030100FB
S12100200A24040A28030201001060230500000000000FFFFFFFFFFFF081500010097
S121003F000900000FF1B3C0050F80F7EE8097EE80C7EE8037EE80652554E2C43503A56
S12100SC310D312C53455442C435036383A310D0000000000000000000000000014
S12100700000000000000000000000000000000000000000000000000000000064
S12100980000000000000000000000005D005E2020202020000000A40000001532
S121009B68B00C44350363820000101000A4004350202020000708000A5002A2A2A2A99
S12100D4FF0D1000000000000000000000000000000000000000000000000000EE
S12100F200000000000000000000000007E07C130EE054FE600C13523037EE113C0
S11201105849BD0030EB01A900CB408903A700AB
S1210111FE701EF00EB01A90030A700E701A60206EE003131AD00306C0626026C053B73
S121013D343434343434C60930A605A700085A26F8C60530A602A709085A26F83930C6C9
S121015B05A609A702085A26F8C609A603A708095A26F831313131313930A605E60608
S1210179A704E7033930A604A705A603A7063930A60536E6068DEF3220E634343086B2
S121019709E602E700084A26F830A605A709A606A70A3930A609A705A60AA706860966
S12101B5E608E70A094A26F831313930BDC68D038DC230A603E604AB06A706E90507C5
S12101D3E7056D0627028AFBA7023930A604C60020E730A60320F7308D9C8D032098E9
S12101F130E605A606A003A706E20420D430E604A60610A706E605C20020C630E603CC
S112020F20F08D3837163230203B38D3030E703AE
S121021EA704075D20B3861036304F5F660666072404EB04A905465666066066076A00B5
S121023C26F03130E703A70407EA05AA0620D78608364F30E606562402AB0746566A94
S121025A0026F6313930A604E603EE053736 3F0534306601 6D012B0B4C6802691 2BBB
S12102780481 1126F5A700A603E6046F036F04E002A2012407EB02A9010C20010D69AA
S1210296046903640166026A0026E631313130080 8A705F7063233A704E703075D7E4B
S12102B401D730E603EE09A60030EE0BA700305A270E6C0A26026C096C0C26E96C0BFA
S12102D220E53930E603EE09A6008104271F30EE0B37E600C104332714A100261030CB
S12102F06C0A26026C096C0C26026C0B5A26D93007A702E7033930E603EE09A600812F
S112030E0427EE30EE0B37E600C1043327E381FA
S12103D3F2704A10026DB306C0A26026C096C0C26026C0B5A26D520C930EE09A6004D
S12103B30EE0BA700308104270E6C0A26026C096C0C26E86C0B20E43930E603E70495
S1210359C10C230630C602E70339EE0B8620C608A700085A26FA862EA70008C6038629
S12103720A700085A26FA306F03EE095FA600812E2708085CC10926F420CB5C30A63A
S12103950410A70427C2810322BEEE09A600812E27B6812A262C30EE0BC608863FA7B1
S12103B300085A26FA308601A703A60A8B02A70AA6098900A709A60C8B08A70CA60BC6
S12103D18900A70B2034C60830EE09A600306C0A26026C09EE0B812E2715A700305A88
S12103EF6C0C26026C0B813F26E0860130A70320D930EB0CE70CE60BC900E70B30EED1
S112040D09A600812A261430EE0B08C603863F89
S121041CA700085A26FA308601A70339306C0C26026C0BE60430EE09A600306C0A2631
S121043A026C09EE0BA700306C0C26026C0B813F26048601A7035A26DE39FCE9FD03AB
S1210458FD1CFD24FD2CFD35FD4AFD5EFD63FD78FD7DFD80FD85FD90FD9CFDA7FDB086
S1210476FE40FE5D00AF02EE02FF02F002F102F202F302F402F502F602F702F802F99A
S12104940 2FA02FB02FC02FD02FE02FF0300030D030E030F03100311 0312FE88FD61F9
S12104B202F102F202F3FDA702F2FE98FE48059C05E78008069305998010059906CB8D
S12104D08010075E07610000059970380020764076700000059C05E78010059959951
S12104EE0000434F4E04C004C0505455204C604C650545004CC04CC44534B04D204D2D8
S112050C4C505404D804D84D544104DE04DE543A
S121054B545904E404E44E554C04EA04EA0030EE058680A705BD054025043F04AD0086
S1210539867FAA05A705393F0508083F05CE04F03F05C6033F1227263F063F0A08086A
S1210557080 83F023F063F063F0508083F053F036D0026DE3F063F060D8605A405A7EA
S1210575053 93F06EE003F023F063F063F05A700E7016D062A04CB0289003F03EE002E
S12105933F023F060C393F04393F033F05EE07862EBD067UBD06688810A27F97D0040FD
S12105B12603BD067D913A26043F0620E0913926163F023F06A1072604E1082704C053
S12105CF0182003F053F0320CFA7000881 0D26C8860ABD067D3F06393F033F05EE071E
S12105EDD63DA60081042770BD067D085A2618860DBD067D860ABD067D8600D63E27D3
S112060B06BD067D5A26FAD63D09A60008810DC4
S121061A26D3860ABD067D8600D63F2706BD067D5A26FA7D003B27217A003C261C96E6
S12106383B973C7D00422607BD0668914326F9D6412708860ABD067D5A26F83F063F76
S12106560 5EE00EE04A600472405A601BD06683F06393F0530EE04EE00EE04A6004704
S1210674FBA601847F3F06393F0530EE04EE00EE0437E600575724FAA701333F06CE
S1210692393F033F05EE078611BD06EDBD06B8810A27F9810027F5A70008810D26EE37
```

```
S12106B08613BD06ED3F06393F0530EE04EE00EE04A6004724FBA6013F06393F033F69
S12106CE05EE07A60008BD06EB810D26F6860ABD06EDC6048600BD06ED5A26F83F060B
S12106EC393F0530EE04EE00EE0437E600575724FAA701333F06393F033F05EE07D6DA
S112070A46A60008BD07485A260C860DBD0748B1
S12107198604BD0748D64609A60008810C260637D644D74533810D26D9860ABD0748CD
S12107377A00452609D644D745860CBD07483F06393F0530EE04EE00EE0437E600A756
S12107550OE6012AFC333F06397E1D867E1DF17EFFFF7EFFFF7E19607E1AF17E1D42C2
S12107737E18CF7E1BA67E1C417E19007E19307E171F7E16097E0B957E07C47EFFFF53
S12107917EFFFF7EFFFF7EFFFF7EFFFF7E11277EFFFF7E0C327E0BBF7E0B587E1D5A4A
S12107AF7EFFFF7EFFFF7EFFFF7EFFFF7EFFFF7E184E7E0A647E0AE17E04C07E04F0CF
S12107CD04D244534B00FF07E700000A000000000000000000000000000000000004B
S12107EB000000000000000000837004D4F564552202020E42494E00010222142262
S11208090700OE000000000000000000004285
S12108184F4253202020202E5852460003002210221000010000000000000000000D4
S121083600AA41434B202020202F535953000100231600000000000000000000000A0
S12108540000000000000000000000000000000000000000000000000000000082
S12108720000000000000000000000000000000000000000000000000000000064
S12108900000000000000000000000000000000000000000000000000000000046
S12108AF0000000000000000000000000000000000000000000000000000000028
S12108CC0000000000000000000000000000000000000000000000000000000A
S12108FA0000000000000000000000000000000000000000000000000000000EC
S11209080000000000000000000000000000DC
S12109170000000000101350135010101350D0CD0101010101013501350101010135013357E
S12109350000000101350135010101350D0CD0101010101013501350101010135013357E
S12109530101013506D081E021D000539013501350D0D71010DA1C9A0135D0A9A9105F
S12109716D106410290B564449521 3A65049500DE152454E11B244454C0DDC5341562A
S12109850F754153530DF04C4F4111174A554D0DC8494E4912BA455849E0E3534543E9
S12109AD0DF6534554 0DFB535441 0DF5424F4F0DFA4C494E0DFF5355420E8700000012
S12109CB000004000000000081040404040442424242424242424242040400040008150
S12109F900828282828282808080808080808080808080808080808080808080040004D8
S11200070000534156452044A41434B2F53595347
S12110A162C2430303230 2C2431453832 2C2430313030 0D0000000000000000000005E
S12110A340000000000000000000000000000000000000000000000000000000000A0
S12110A52000000000000004C0434F4E00000A0900008E0977F0EC8FE07C8C6043F92
S12110A700AFE008603A7008615A700FE07C8C6283F0AEE008603A7008601A700FE07A5
S12110A8EC8C61C3F0AEE0086FFA700862CA7013F33CE0A598643A702864FA703864E77
S12110AACA704CE0ABF3F31CE0AC13F31CE0000DF2920320A0D0A48454D454E574159C6
S12110AC204153534F4349415445532043502F3A382D312E300A0D8E09777F0EC8DF97
S12110AF8292706 3F15EE2526FACE0000DF2BDF2DDF2FDF313F30D625C101270BC103EB
S11208062720CCE0B6B3F3120EDDE203F05CE09AF
S12110B15783F05C6033F12272C3F063F0A08083F023F06DE203F053F036D0026E43F37
S12110B3306DF20DF23BD1117CF106D6D0526AFDE2727ABAD0020A73F06EE003F023F25
S12110B51063F03AD00209930A605E606CE0A59A707E7088680A7063F1339464F524DCD
S12110B6F4154204552524F520D4E554D424552204552524F520D20000000204552526F
S12110B8D4F523A2000000A0D30EE05A6052722BD15FAB70B91A605BD15FEB70B92A689
S12110BAB02B70B86A603B70B87A604B70D88CE0B853F3139CE0A596F068643A7028653
S12110BC94FA703864EA7043F02CE0A093F04A707E7087D0EC8274BCE0A09FF0EC9CE46
S12110BF70ECB3F18E605C108270B502712BD0EA87E0EC820C3CE0ECB3F157F0EC82025
S1120C05B9FE0EC9A70008FE0EC9810D2609CE3E
S12110C140A093F31DF232016812676056F0A5920068104270320BB3F13CE0A09DF234B
S12110C327F00227C0022DE23DF20A60008DF23812027F3220781 0D263A162032815E97
S12110C5023022031BD0D118580270DC6018501270 2C602BD0CB220188540270 5BD0C4D
S12110C6E80200F8504271 0DF20E600C1242603BD0CD49726D725397F00267F002539E6
S12110C8CDF23A6007C002208DF23BD0D11854026EFD6225AD722C1062D037E0D07BDB1
S12110CAA0D7CDF27C603704DDF23A6007C002208DF23BD0D1185012702C60285802697
S12110CC8E9854026E596224A9722202B7E0022DE23DF20DE23A6007C002208DF23BD9E
S12110CE60D11850226EFD6225AD722C1052D037E0D07BD0D25DF27C603DE2309DF2395
S1120D048602 9CE0B783F3131317E0C85812048
S12110D13250C815E2208CE09A93E09A600394F390100DE207F0D237E0D24D6220908ED
S12110D315A26FCD622BD0D6CB70D245A272909BD0D6C48484848BA0D24B70D245A27B1
S12110D4F1809BD0D6CB70D235A270E09BD0D6C48484848BA0D23B70D23FE0D2339A678
S12110D6D0080308 1092F028007390000000A7F0D777E0D787E0D7A7E0D7B7C0D7B17
S12110D8BDF2009D622F70D79085A26FC3F05E600C40F4FFE0D7A3F103F02FB0D78B9AC
S12110DA90D77B70D77F70D784FC60AFE0D7A3F10FF0D7A3F06097A0D7926D3FE0D77AB
S12110DC7393F2D625C1032706CE0B6B3F31393131DE276E00CF0E272021CE0E342041
S12110DE51CCE0F3E2017CE0E4D2012CE0E57200DCE0E642008CE0E712003CE0E7C3F55
S1120E0302DE233F053F03DF23BD11173F06DF48
S12110F1223CE106D6D05260CDE272708B607CBF607CC6E0039303A44454C4554452E35
S12110F30434D440D303A5049502E434D440D303A534543555249545 92E434D440D303C
S12110E4E30 5345542E434D440D303A41535349474E2E434D440D303A535441545553C1
S12110E6C2E434D440D303A424F4F542E434D440D303A4C494E4B2E434D440DCE0ECB5A
S12110E8A6F0686EFA7293F2C3F1E6D05261 03F146D05260AA61D810326047C0EC8391B
S12110EA8CE0EB53F31CE0ECB3F1E3F153920535542 4D49542046494C4520455524F0A
S12110EC6520D0000000000044534B00000EF500000000000000000000000000000C6
S12110EF4000000000000000000000000000000000000000000000000000000000EC
S1120E020000000000000000000000000DC
S12110F1100000000000000000000000000000000000000000000000000000000BE
S12110F2F0000000000000000000000000000000000000000000000000000000A0
S12110F4D0000000000000000000CE106D6F1D6F296F1E86EFA7063F2C3F1E6D0527D5
S12110F8901393F2F962681042706CE0B6B3F31393F2FD625C10326F2DE27FF139E3F0A
S12110FA72F962681042 6E53F2FD625C10326DDDE27FF13A03F2F962681 0D272B810437
S12110FC526CC3F2FD625C10326C4CE106D6C1D3F146D0527037E105A8616BD106296F5
S12110FE327BD10629628BD10622009CE106D3F146D052663B613A0F613A1F0139FB280
S1121001139E2729C6FFCF106D86028D54B61399
S12110109E8D4FB6139F8D4A178D47FF139EA60008FF139ECE106D8D395A26EF20C9A4
S1211102ECF106D86028D2DB6139E8D28B6139F8D23178D20FE139EA60008FF139ECE40
S1211104C106D8D125A26EF3F156D0526 0139CE106D3F1E3F15393F196D052704313145
S1211106A20EE3904D244534B00FF1097002410241924074A41434B202020202E5359B0
S1211088530001002316000000220000 10DF00241924072 6EF3F156D0526 0139CE1027
S12110A66D3F1E3F15393F196D0527043131 20EE3904D244534B00FF1097002410247D
S12110C41924074A41434B202020000000000000000000000000000000000000004D
S12110E2000000000000000000000000000000000000000000000000000000000EC
S1121100000000000000000000000000DC
S12111 0F0000000000000000CE106D3F2C3F1E6D0526043F253F1E397F00277F0028C8
S12111 2D30EE056F066F296F053F146D05270139A61D2722810127 1ECE11503F313034
S12111 4BEE053F153920494C4C4547414C2046494C452D545950450D30EE058D3981C2
S1211169162 60A8D33972 78D2F972820FD1810227033F15398D22B7139E8D1DB7139FB4
S12111878D18B713A530EE058D10FF139EA70008FF139E7A13A526E020C23F18E605FB
S12111A5270AC10826026F0531313F1539CE106D6F063F2C3F1E6D052628BD16BFCEFB
S12111C307CDA60AE60BA71FE720A627E628A721E722EE27A60E810226037E12A4CEA5
S12111E1106D6D052716CE11FD3F3139205245 4E414D494E4720455252 4F520D3F2FB5
S11211FF962681042706CE0B6B3F31393F2FD63E
S121120E25C10126F2DE20FF139E9622B713A53F2FD625C12F26E07C13A53F2FD625EF
S121122CC10126D5D622FB13A5CE107D3F05FE139E3F053F3431313131 5D26BDCE1051
S12112 4A6DBD16BFCE106DA60526037E128E81027037E11F7CE07CD6F05A61FF62043
S12112 68A70AF70B3F134D26FBEF213F05CE107D3F05C60C3F1131313131CE07CD8611
S12112 86FFA7063F136F0639CE12943F3139204555504C494341544 5204E414D450D74
S12112A4CE12AA3F3139205345435552495459204552524F520D3F2FD625C103270C46
S12112C29626810427F2CF0B6B3F31397D0027262096288103271A36CF12F9DF23BD8D
S12112F0111732CE106D6D052608DE27270497286E0039CE0B783F3139303A494E49CD
S11212FE542E434D440D4449524543544F5259C5
S121130D204F46204452495645203 00A0D202020204E414D4520202020202020542013
S12113B2020412020465 42D46532020 4C542D4C532020204F530A0D424F4253202055
S12113492020 2E5852462020303320203030202032322031302020323220313020202057
S12113673030303 10D037F3F3F3F3F3F3F3F2E3F3F3F002A2E2A0A303038393 52E71
S12113852053454354 4F525320555345440A0D271003E80064000A0001139F1E8200B8
S12113A30000057F13A57F13A4860CB7139ECE136E3F05CE137B3F05C6033F343131EB
S12113C131313F2FD625C10D26037E1455C12F260F3F2FDE20A600814C2637C13A4CD
S12113DF20E2C10326267D0027260B9628810322 05B713A52006CE0B783F31393F2F9F
```

```
S11213FDD625C10DZ752C13A26FE3F2FD625C162
S121140C01270AC1022706CE0B6B3F3139DE20FF13A09622B7139E3F2FD625C12E2661
S121142AF87C139E3F2FD625C1012706C1022702220D7D622FB139ECE136E3F05FE1308
S12114480A03F053F343131313C10227BF7F136C7F136D7D13A4270FCE0A59864CA7AE
S12114660286504038654A704B613A58B30B71317CE13043F31CE131A3F31CE07CDF1
S12114843F02CE07F73F04A707E708B613A5A7093F17A605274B81012644B6136CF6B1
S12114A2136DCE137FFF139ECE13947F13A5E001A2025057C13A520F5EB01A900362B
S12114C0FF13A0FE139EB613A58B30A7003208FF139EFE13A008088C139E26D1CE1319
S12114DF7E3F31393F1E39EE27A600812026037E15F23F05CE136E3F05C60C3F3531D7
S12114FC31313127037E15F2CE136AC6208627BD
S121150BE700094A26FA3F05CE07CDEF273F05C60C3F113F063F060808083F05CE0748
S1211529CDEE27A60D16BD15FA3F06A7000817BD15FEA7000808083F05CE07CDEE2794
S1211547A60E16BD15FA3F06A7000817BD15FEA7000808083F05CE07CDEE27A60F168C
S1211565BD15FA3F06A7000817BD15FEA70008083F05CE07CDEE27A61016BD15FA3F34
S12115830 6A7000817BD15FEA7000808083F05CE07CDEE27A611168D5E3F06A7000844
S12115A1178D5AA70008083F05CE07CDEE27A61168D463F06A70008178D42A700084E
S12115BF08083F05CE07CDEE27EE133F02FE136C3F08FF136C3F06368D21A700083271
S12115DD8D1FA70008178D15A70008178D13A700CE13433F31CE07CD3F1A7E149644D0
S11215FB444444840F8B3081392302 8B0739CE4B
S121160A106D3F0530EE073F05C61F3F113131313 1CE106D3F02CE10973F04A707E7C3
S1211628086F053F256D05260FDE27270B3131313131313131316E00CE16783F05CE4D
S1211646107D3F05C60C3F113131313 1CE16663F31313131313131313131CE106D3FCE
S121166415392055 4E41424C4520544F20434841494E3A2000000000000000000000009F
S12116820000 0A0DCE07CDA7093F02CE07F73F04A707E7086F053F17A605270781 01CB
S12116A027137E14E2EE27A600812026013 9CE07CD3F1A20E5A60B811A2601396F059E
S12116BE393F05A609CE07CDA7093F02CE07F73F04A707E7086F053F17A605270B8177
S12116DC01273B3F06A7057E14E2EE27A600812026 07CE07CD3F1A20E23F0530EE023A
S11216FA86103F093F05C60C3F123131313126AE
S1211709E4CE07CDEE273F023F06A727E7286F05393F06A7053930EE058D9B30EE057B
S1211727 6D05270139EE276D0E2724CE173F3F3130EE05 8612A705392046494C45205E
S1211745444454C4554452D50524F544543544544 0DDE2927466D06270DA6093F0530AD
S1211763FE07A10927083F06EE2526EB202F3131CE17803F31861230EE05A7053920E7
S1211781444454C4554452045524F522D4F50454E204F5554505554204649 4C4553E6
S121179F0D30EE05EE27A60FE61030EE05A71FE720EE27A611E61230EE05A721E72290
S12117BDEE278620A700CE07CD86FFA7063F136F066D05270139A609CE106DA709860A
S12117DB00C603A70AE70B6F063F13A60526E4EE07A67EE67FB7139EF7139F30EE0552
S11217F9A61FE620CE106DEE07A77EE77F36CE43
S1211808106DA60948CE002B3F0932A700E701CE106D63063F136F06A60526AA30EE34
S12118 2605A621E622CE106DA70AE70B3F13A6052696EE07B6139EF6139FA700E70192
S1211841CE106D63063F136F063930EE056F056F093F2FD625C10326297D0027260A6F
S12118629628810322 04A709200E30EE058615A7057F00277F0028393F2FD625C13ACA
S12118 8026EA3F2FD625C101270830EE058615A70539DE20FF139E9622B713A53F2FF6
S121189ED625C12E26E67C13A53F2FD625C10126DBD622FB13A530EE0586103F093FE7
S12118BC05FE139E3F053F34313131315D26BF397E07CD30EE056F066F0A8604A70BC1
S12118DA6F05A607E608A727E7283F134D260CEE276D00270930EE056F0539A70539C8
S11218F830EE058601A7053930EE05A627E62850
S1211907CB208900A727E728EE073686803F09323F0B270830EE054FA70520C630EE27
S1211925 05A60B4C811B27CB7E18D830EE05EE273F0530EE0786103F093F05C6153FC5
S1211943113131313 1FE18CD63063F1330EE05A7053900807E16867E16BF7E07CD3098
S1211961EE053F02DE2927153F053F0C3F062706EE2526F420073F03860DA705393F9F
S121197F036D0627037E1A19BD195A6D0527058604A70539861D3F093F0530EE07EE76
S121199D27860D3F093F05C6083F113131313130EE05A61FE620A70AE70B3F134D27A9
S12119BB03A70539EE07A600E60130EE05A70CE70DEE07A602E60330EE05A70EE70F82
S12119D9A607E608CB048900A727E7286F256F26DE29260E30A605E6069729D72A3FF1
S11219F7036F05396D2526166D2626123F053020
S1211A06A607E6083F06A725E7263F036F0539EE2520E2BD195AA60581012705860 3EF
S1211A24A70539A609BD1957FE195EA605270E810127063 0EE05A7053986062 0F6A68B
S1211A4227E62830EE05A727E7286F236F246F216F226F0E6F0FA609840348CE002B95
S1211A603F093F05A600E6014D27035D262730EE076F068600C603A70AE70B3F1363E9
S1211A7E064D2705A705313139EE07A67EE67F3F063F05A700E70130EE074D27035DF1
S1211A9C26078607A705313139A71FE720A70AE70B3F1B6D0527033131396F063F135F
S1211ABA63064D2705A705313139EE07A600E6013F064D260B5D260886073 0EE05A78A
S1211AD80539A700E70130EF05FF07C67C6F04085A26FA30EE057E19D930A605E60676
S1121AF6DE29272E3F053F0C3F06260AA625E6CC
S1211B05269729D72A2023A1252614E12626103F053F03A625E6263F06A725E72620BC
S1211B230BEE2526E43F03860DA7053930EE056D062601396D0A27046D0B260AA60ECA
S1211B41E60FA70AE70B200E3F13A623E624CB018900A723E7246F06BD195A30EE05A5
S1211B5F63066D05270139A60AE60BA721E7223F1B6D052701396F068600C603A70A14
S1211B7DE70B3F1363064D270139A609840348CE002B3F09A600E60130EE05EE07A7E0
S1211B9B7EE77F30EE053F13A7053930EE056D062705 612A70539A627E628E008A246
S1211BB907B119552605F119562736EE27A6002A1E30EE056D292715EE274C260486F3
S1211BD720200EA700862030A704EE056F0539EE27083F0530A706323330EE05A72742
S1121BF5E7286F0539A60AA121260BA60BA1220A
S1211C042605 8608A70539A60CE60DA70AE70B3F134D26F0A607F608CB048900A72762
S1211C22E728EE07A600E60130EE05A70CE70DEE07A602E60330EE05A70EE70F7E1B58
S1211C40C430EE056D062605 612A70539A627E628E008A207B119552605F119562743
S1211C5E4530A604EE056D292727847F81202621EE27F6002A0B5A2A0CE70030EE05B9
S1211C7C6F0539C6FF20F4083F0230EE05A727E72820C2EE276D0026EEA700083F020F
S1211C9A30EE05A727E7286F0539A6098403CE002B483F09A6002718E601271430EE97
S1211CB805EE07A700E70130EE053F134D2706A705397E1D3AA623E624CB018900A709
S1211CD623E724A60AE60BA70EF70FA6098403CE002B483F09A600E6013F0530EE07BD
S1121CF4A70AE70B6F063F1363064D2705A705E5
S1211D03313139EE07A600E6013F06A700E70130EE05A70CE70DA607E608CB04890010
S1211D21A727E728A60EE60FEE07A702E703C67C6F04085A26FA7E1C5F860730EE05B2
S1211D3FA7053930EE056D062705 612A705396F053F156D052701393F14397FEC01CB
S1211D5D7FEC037FEC077FEC0086FFB7EC02B7EC068604B7EC01B7EC07862CB7EC0316
S1211D7B8680B7EC02860CBD1E4D393F03A6090C464646AA0BE60AFE07BD1E41BD1E48
S1211D9974240220 4917BD1E66C6058602BD1E4DB6EC008508270ABD1E5E5A26EE86C5
S1211DB705202D858027048609202 5C680863CB7EC038640B7EC02B6EC0036862CB75A
S1211DD5EC038640B7EC027FEC0232A700085A26DE4F30A706EE07AA05A705393F03EF
S1121DF3A6090C464646AA0BE60AFE073637C683
S1211E0280A60008B7EC068630B7EC025A26F23332BD1E41BD1E74240220CA17BD1E48
S1211E2066C6058604BD1E4DB606BD1E4DB6EC008508270ABD1E5E5A26E9860520A9B3
S1211E3E4F20A6BD1E5EB7EC068620B7EC023936B6EC0032B7EC02B6EC012AFBB6ECEE
S1211E5C003936860AB7EC023239B7EC068610B7EC028608BD1E4D39B6EC0084202616
S10B1E7A034F0C390D860A39FF
S9
```

```
LOAD LIMITS:
0100 1E81

LOAD MAP
BIOS   0100
EQTAB  04C0
PDTAB  04F0
.MTIN  0765 UNRESOLVED
.MTOT  0768 UNRESOLVED
@USR6  078F UNRESOLVED
@USR7  0792 UNRESOLVED
@USR8  0795 UNRESOLVED
@USR9  0798 UNRESOLVED
@USR10 079B UNRESOLVED
@LOADR 07A1 UNRESOLVED
@USR1  07B0 UNRESOLVED
@USR2  07B3 UNRESOLVED
@USR3  07B6 UNRESOLVED
@USR4  07B9 UNRESOLVED
@USR5  07BC UNRESOLVED
@CLDST 07C1
CLI    07C1
@WRMST 07C4
SYSFCB 07CD
@PRTMS 0B58
@PRTER 0B95
@GTCMD 0BBF
@NXTOK 0C32
CVHB   0D25
CVDB   0D7C
@LOADR 1127
@CHAIN 1609
SEMPTY 1686
SFILE  16BF
@DELET 171F
@FMTFC 184E
DIRECT 18CC
@OPENU 18CF
@GETDR 1900
@PUTDR 1930
SFIO   1955
@OPEN  1960
@CLOSE 1AF1
@READ  1BA6
@WRITE 1C41
@REWD  1D42
@INTDK 1D5A
ICOMDR 1D5A
.RDSEC 1D86
.WTSEC 1DF1
```

ASSIGN

```
S0032000DC
S1212000B7211BF7211C3F2FDE20A600914326093 9CE20F13F317E207ED625C10126FB
S121201EF2D622C10326ECCE211D3F05DE203F05C6033F11313131313F2FD625C13D0A
S121203C26D33F2FD625C10126CBD622C10326C5CE21203F05DE203F05C6033F1131E7
S121205A313131CE211D3F05CE21203F05C6033F123131313126 25CE211DBD20BB243D
S12120781 3CE20FE3F31CE2112 3F313F30DE20DF237E2006A602A700A603A70120E8AB
S12120 96CE2120BD20BB240220D9FF023F05CE211DBD20BB240220CB3F023F063F04B0
S12120B4A700E7017E207E3F05FE211B3F05C6033F1227233F063F0A080808083F0547
S12120D230FE028603103F0D3F0230A702E7033F066D0026D931310D393F0631310CD7
S11220F03953594E54415820455 524F520D49BD
S12120FF4E56414C4944204445564945204E414D450D41535349474E2D20040000CD
S106211D000000BB
S9
```

BOOT

```
S0032000DC
S12120008E400F7FEC017FEC037FEC077FEC0086FFB7EC02B7EC068604B7EC01B7EC86
S121201F07862CB7EC038680B7EC02860CBD21478603C600CE4010BD20DDCE4010A6F4
S121203C7AF67BB74090F74091A67CE67BB74092F74093CE4014FF4096B64091F640CC
S121205A90B74095F74094CE4010BD20DD8D3A8116260C8D34B7409A8D2FB7409B2055
S1212078EE810226218D24B740988D1FB740998D1AB7409C8D15FE4098A70008FF4072
S1212096987A409C26F020C9FE409A6E00FE40968C40902707A60008FF409639F640A5
S12120B494B64095B140932607F1409226022 0DACE4010E600A601F74094B740958DC6
S12120D20ACE4014A60008FF4096393686 0AB7EC0232B7EC068620B7EC02B6EC00844D
S11220F02027037EE11317B7EC06861 0B7EC0226
S12120FF86088D44C60586028D3EB6EC008508270B860AB7EC025A26ED7EE1138580C8
S121211D27037EE113C680863CB7EC038640B7EC02B6EC0036862CB7EC038640B7EC52
S121213B027FEC0232A700085A26DE3936B6EC0032B7EC02B6EC012AFBB6EC0039CE7B
S10A2159215E3F3139200A29
S9
```

DELETE

```
S00320B824
S1212000000044534B0000202A0000000000000000000000000000000000000000000092
S121201E0000000000000000000000000000000000000000000000000000000000000000A0
S121203C0000000000000000000000000000000000000000000000000000000000000082
S121205A0000000000000000000000000000000000000000000000000000000000000064
S1212078000000000000000000000000000000000000000000000000000000000000046
S1212096000000000000000000000000000000000000000000000000000000000000028
S12120B400000000CE20006F096F063F2FDE20A600914326 139D625C10326347D0053
S12120D227260D962881032207CE2000A7092016CE20EA3F317E2221204E554D4245B3
S11220F052204552524F520D3F2FD625C13A264A
S12120FFE23F2FD625C101271AC1022716CE21143F317E222120464F524D41542045EF
S121211D52524F520DDE20FF202A9622B7202C3F2FD625C12E26D87C202C3F2FD625C5
S121213BC1012704C10226C9D622FB202CCE20AC3F05FE202A3F053F3431313131C142
S12121590227B0CE20006F293F17A6052728810126 1F6D2927037E2221CE217A3F3194
S12121777E22212046494C45204E4F5420464F554E440D3F1E7E2221EE27A600812011
S12121952711 3F05CE20AC3F05C60C3F353131313127 07CE20003F1A20B4CE20006C21
S12121B329CE222F3F31CE2000A6098B30B722 39CE22393F31CE2000EE278604A70C0F
S12121D13F31CE223C3F313F30DE20A600815926C6CE20103F05CE2000EE27FF20AAF9
S11221EF3F05C60C3F1131313131CE20003F1C6A
S12121FEB620AAF620ABA727E7286D05269CCE22133F312095204649 4C45204454C10
S121221C455445440DCE222F3F313F30DE20DF237E20B8204454C4554452D200400F9
S10C223A3A04203F2004111A09A2
S9
```

INIT

```
S00320C319
S1212000000044534B00FF000000000000000000000000000000000000000000000000DD
S121201E000000000000000000000000000000000000000000000000000000000000000A0
S121203C000000000000000000000000000000000000000000000000000000000000082
S121205A000000000000000000000000000000000000000000000000000000000000064
S1212078000000000000000000000000000000000000000000000000000000000000046
S1212096000000000000000000000000000000000000000000000494E49542E204449534B7B
S12120B420494E2044524956452000203F20049628840 3CE2000A7098B30B720BECE15
S12120D220AA3F313F30DE20A600815927013 9CE20006F0A8601A70B3F02CE22443F10
S11220F004A707E7088D466D05270220136C0B24
S12120FF3F02CE22C43F04A707E7088D316D052702207A6C0B3F02CE202A3F04A70736
S121211DE7083F05CE202AC67E4FA700085A26FA8601A700A7013F068D7E6D052704D6
S121213B204F20766C0B7F20A87F20A98D6C6D052702203DA60B4C811A2704A70B20F1
S1212159ED8601A70BA70A165CC11B2609C6014C814D26024F5FB7202A378D33F7204A
S12121772B338D3A4D26045D260139A70A378D21E70B3320D5CE21923F3139494E4933
S1212195549414C495A4154494F4E204641494C45440D3F05CE222A3F0A09F6003F39
S12121B30639364F6F053F13A7054D2602323916 8D55B7220017 8D53B722013F05A668
S12121D10B8D46B7220DA60B8D43B7220EA60A8D38B72217A60A8D35B72218CE21F514
S11221EF3F313F1F39394449534B204552524F1A
S12121FE523A00002041542053454354 4F522000002C2054524143 4B2000000D4444F8
S1212210444840F8B3081392302 8B073900010A13020B14030C15040D16050E170666
S10F223A0F1807101908111A0912464366
S9
```

LINK

```
S00320AA32
S1212000000044534B0000202A0000000000000000000000000000000000000000000092
S121201E0000000000000000000000000000000000000000000000000000000000000000A0
S121203C0000000000000000000000000000000000000000000000000000000000000082
S121205A0000000000000000000000000000000000000000000000000000000000000064
S1212078000000000000000000000000000000000000000000000000000000000000046
S1212096000000000000000000000000000000CE20006F096F06CE21CE90
S12120B43F313F30CE2000D625C103262F7D0027260A962881032204A7092014CE201E
S12120D2D63F3139204E554D4245522045525 24F520D3F2FD625C13A26E43F2FD625F6
S11220F0C1012714CE20FA3F31920464F524DFB
S12120FF4154204552524F520DDE20FF202A9622B7202C3F2FD625C12E26DA7C202C51
S121211D3F2FD625C10126CFD622FB202CCE20103F05FE202A3F053F34313131315DDF
S121213B26B7CE20003F17A605271D81012616CE21503F3139204649 4C45204E4F547B
S12121592046 4F554E440D3F1E39EF273F05CE20103F05C60C3F12313131312707CFA7
S12121772000 3F1A20C5CE2000EE27A60FE610CE2000A71FE720EE27A611E612CE20CD
S12121950 0A721E7228600C603A70AE70B3F136D0527037E2160CE20006306A61FE676
S12121B320B720A4F720A5A621F622B720A6F720A73F136D0527037E2160392053597B7
S12121D1535445 4D2046494C45204E414D453F2004B72217A60A8D35B72218CE21F598
S10421EF3FAC
S9
```

S00324580
S1212000000000000000000202C0000000000000000000000000000000000000000000000000072
S121201F000000000000000000000000000000000000000000000000000000000000000000A0
S1212038000000000000000000000000000000000000000000000000000000000000000082
S1212050000000000000000000000000000000000000000000000000000000000000000064
S1212078000000000000000000000000000000000000000000000000000000000000000046
S1212096000000000000000000000000000000000000000000000000000000000000000028
S12120B4000000000000000000000000000000000000000000000000000000000000000A
S12120D2000000000000000000000000000000000000000000000000000000000000000EC
S11220F000000000000000000000000000000DD
S12120FF000000000000000000000000000000000000000000000000000000000000000BF
S1212110000000000000000000000000000000000000000FF215800000000000028
S1212138000000000000000000000000000000000000000000000000000000000000000082
S1212159000000000000000000000000000000000000000000000000000000000000000064
S1212177000000000000000000000000000000000000000000000000000000000000000046
S1212195000000000000000000000000000000000000000000000000000000000000000028
S12121B3000000000000000000000000000000000000000000000000000000000000000A
S12121D1000000000000000000000000000000000000000000000000000000000000000EC
S11221FF000000000000000000000000000000DD
S12121FE000000000000000000000000000000000000000000000000000000000000000BF
S1212210000000000000000000000000000000000000000000000000000000000000000A0
S1212238000000000000000000000000000000000000000000000000000000000000000082
S1212258000004F4E222DE2321233023585B5545222DE2321233000050545022DE23CB
S121227621000023584453422B322BF22C422D44C50542388238F0000235B4D544186
S121229422DE2321233023585B54545922DE2321233023584E554C22DE232100023584C
S12122B2003F033F146D052601393F1E393F033F1539CE20003F18E605C108260139E5
S12122D05D26E939CE212C3F196D0526DF393F036D062624FE202AEE07260DCE23A71D
S11222EF3F31CE20008612A70539CE20003F13C2
S12122FDA607E608A727E7286F0539FE2156EE09260DCE23BD3F31CE212C8612A7057E
S1212318B9CE212C20DC3F036D062601398604BD2F860D202BCE2000EE27A6008104E9
S1212339260BCE2000C608E7053981 0D270F08363F02CE2000A727E728326F05398DF9
S1212357A0860D39CE212CEE27A700088C22582713363F02CE212CA727E72832CE2149
S12123752C810D278139CE23D43F31C612CE212CE70539BD22DE860C20CC3920424147
S1212393442049 4E5055540D2042414420 4F55545055540D20494C4C4547414C20499F
S12123B14E505554204445564943450D20494C4C4547414C204F55545055542044451 2
S12123CF564943450D2042554646455220 4F5645522D52554E0D204E554D424552204A
S11223ED4552524F520D20464F524D41542045F8
S12123FC52524F520D20494C4C4547414C2053574954434800DDE203F05CE225A3F0584
S1212418FF2455D622C10326233F127725FE2455C60B3F0AFF2455A600271131313100
S1212438310E203F05FE24553F05C60320DD313131310D3931313131FE24550C390004
S121245600000CE227BFF202AFE2156CE20006F29CE212C6F096F297F27987F22597FD2
S1212474225866F058644A7028653A703864BA7048620A7108600A71D3F2FDE20A6002D
S1212492914326013 9D625C1032636 7D0027260D962881032207CE212CA7092008CED6
S12124B023E53F317E26F33F2FD625C13A26F03F2FD625C13D26067327987E2597C1C1
S12124CE012606202 8C1012708CE23F33F317E26F3BD24112517FF2156CE212E3F0596
S11224ECFE21563F05C6033F1131313132035F2
S12124FBDE20FF24559622B724573F2FD625C12E26CA7C24573F2FD625C10126BFD63A
S1212519 22FB2457CE213C3F05FE24553F053F34313131315D26A73F2FD625C12F26FE
S12125373C3F2FDE20A600CE212C814226077C22596F1D20E8143260886 03A71D6CF7
S1212555292 0D8814826097C22588603A71D20CB815426068603A71D20C1CE240120DB
S12125736 4CE212C3F02FE2156EE03AD00CE212C6D052708CE239B3F317E26F3DE2026
S1212591E600C13D2673CE227BFF202ACE20006F096F058644A7028653A703864BA7AF
S12125AF048620A7103F2FD625C1032707 7D279827452047 7D0027260D962881032204
S12125CD07CE2000A7092014CE23E53F31CE212C3F02FE2156EE05AD007E26F33F2F57
S11225EBD625C13A26E43F2FD625C10D26037EFF
S12125FA2719C101272BC10226062025C1012705CE23F320C9BD24112517FF202ACE37
S1212618200 23F05FE202A3F05C6033F1131313131206 DE20FF24559622B724573FCC
S12126362FD625C12E26CD7C24573F2FD625C1012704C10226BED622FB2457CE201016
S12126543F05FE24553F053F34131313 1C10227A57D279827037E289AC1012799CEA9
S1212672200 03F02FE202AEE03AD00CE20003F1E6D052706CE23907E25D8A61D8103D2
S1212692702 7D2258270D7E2B516C297D225927037E2D43FE202AEE07AD00CE200057
S12126AFE605C108271C5D27033F1E4FFE2156EE09AD00CE212C6D0527DC3F1ECE23E9
S12126CC9B7E25D83F02FE202AEE05AD003F2FD625C10D2606CE27137E25D89626818A
S11226EA0426037E25977E260ACE20003F02FE9B
S12126F9202AEE05AD00CE270D3F313F30DE20DF237E2458205049502D0420444F4EBF
S1212717450DCE2000A6098B30B727ABCE212CA6098B30B727B6CE27993F313F30DE09
S1212735 20A600815926598600C601CE212CA70AE70BCE2000A70AE70B3F136D0527DC
S1212753073F1ECE27CB3F31CE21583F05CE202C3F05C6803F1131313 1CE212C3F33
S1212771136D0527073F1ECE27D73F31CE2000A60AE60B5CC11A26B7C6014C814D2650
S121278FB0CE27BB3F317E26FF000A20434F50592046524F4D2044524956452000 2022
S12127AD544F2044524956452000203F20042043 4F505920434F4D504C4554450A0D3F
S12127CB2052454144204552524F520D20575249544520452524F520D0000044534BB5
S11227E90000280E0000000000000000000000A7
S12127F8000000000000000000000000000000000000000000000000000000000000000BF
S1212816000000000000000000000000000000000000000000000000000000000000000A0
S1212834000000000000000000000000000000000000000000000000000000000000000082
S1212852000000000000000000000000000000000000000000000000000000000000000064
S1212870000000000000000000000000000000000000000000000000000000000000000046
S121288F000000000000000000000000CE288E3F05CE20103F05C60C3F113131313138
S12128ACCE2000A609CE27E4A7096F066F293F17A605273C8101261F6D2927037E2648
S12128CAD0CE28D33F317E26D02046494C45204E4F5420464F554E440DCE28EB3F3124
S11228E87E26D02044952454354 4F525920452F
S12128F752524F520DEE27A600812027113F05CE288E3F05C60C3F353131313 1270795
S1212915CE27E43F1A20A0CE27E46C29CE2A043F31CE27E4A6098B30B72A0CCE2A0CA0
S1212933 3F31CE27E4EE278604A70C3F31CE2A0F3F313F30DE20A600815926C6CE2039
S1212951103F05CE27E4EE27FF24553F05C60C3F113131131CE213C3F05FE24553F5B
S1212968E4B62455F62456A727E7287E2915A61DCE212CA71D3F146D05271C3F1ECE37
S1212980E4B62455F62456A727E7287E2915A61DCE212CA71D3F146D05271C3F1ECE37
S1212998E4B62455F62456A727E7287E2915A61DCE212CA71D3F146D05271C3F1ECE37
S12129AB239B3F31CE212C3F15CE27E4B62455F62456A727E7287E2915CE20003F1817
S12129C9E605C108271B5D27073F1ECE23903F31CE212C3F196D0527E23F1ECE239B46
S11229E77E25D8CE20003F15CE212C3F15CE27BC
S12129F6E4B62455F62456A727E7287E291520434F50592D2004003A04203F20040036
S1212A1400000000000000000000000000000000000000000000000000000000000000A0
S1212A3200000000000000000000000000000000000000000000000000000000000000082
S1212A5000000000000000000000000000000000000000000000000000000000000000064
S1212A6E0000000000000000000000000000000000000000000000000000000000000046
S1212A8C0000000000000000000000000000000000000000000000000000000000000028
S1212AAA0000000000000000000000000000000000000000000000000000000000000000A
S1212AC8000000000000000000000000000000000000000000000000000000000000000EC
S1122AE600000000000000000000000000DD
S1212AF5000000000000000000000000000000000000000000000000000000000000000BF
S1212B1300000000FE202AEE07AD00CE2000E60539FE2156EE09AD00CE212CE605394C
S1212B313616FB2A14F72A148D078DE7328D0620E24444444840F8B30813923028B31
S1212B4F07398DC45D27037E2CA48116263E86538DC35D26F286308DBC5D26EB7F2A4F
S1212B6D1486038DBF5D26E18DA05D26DC8DB55D26D78D965D26D28DAB5D26CDB62AF1
S1212B8B14438DA25D26C4860D8D8F5D26BD20B6810227037E2CBCBD2B175D27037E80
S1212BA92CA4B72A15BD2B175D26F4B72A16BD2B175D26EBB72A13B72457CE2A17FF36
S1212BC72455BD2B175D26D9FE2455A70008FF24557A245726ECCE2A17FF2455B62A0C
S1122BE513811E22507F2A148653BD2B245D2694
S1212BF4B38631BD2B245D26ABB62A138B03BD2B315D26A0B62A15BD2B315D27037E4B
S1212C122CA4B62A16BD2B315D26F4B62A13B72457FE2455A60008FF2455BD2B315D12
S1212C3026DF7A245726EC7E2B897F2A148653BD2B245D26CC8631BD2B245D26C486C3
S1212C4E1EB724578B03BD2B315D26B7B62A13801EB72A13B62A15BD2B315D26A6B6C1
S1212C6C2A16BD2B315D269DFE2455A60008FF2455BD2B315D2621FE2A1508FF2A15F0
S1212C8A7A245726E5B62A1443BD2B315D260B860DBD2B245D26037E2BE3C108271495
S1212CA86B0626083F1ECE23907E25U83F1ECE239B7E25D8CE20003F02FE202AEE0542
S1212CC6AD003F2FD625C10D261E8653BD2B245D26CC8639BD2B245D26C4860DBD2B03
S1122CE4245D26BCCE27137E25D87E26E7BD2B84
S1212CF3175D261280302B0F81092F0A81112B0781162E03800739CE2D123F31C6FFD8
S1212D1139204241442048455820434841524143544552 0D8DCA5D26184848484 8B7F8

```
S1212D2F2A178DBE5D260CF62A171B16FB2A14F72A145F39BD2B175D27037E2E338113
S1212D4D5326F3BD2B175D26F38130265F7F2A148DC65D26E78DC15D26E2B72A158D9D
S1212D6BB95D26DAB72A16B62A1443B724578DAA5D26CBB124572716CE2D897E25D8E3
S1212D892043845434B53554D20455524F520D8616BD2B245D27037E2E33B62A15FB
S1212DA7BD2B245D26F4B62A16BD2B245D26FB7E2D438131266F7F2A14BD2D255D2693
S1212DC5DB8003B72A13BD2D255D26D0B72A15BD2D255D26C7B72A168602BD2B245D01
S1122DE326BCB62A15BD2B245D2645B62A16BD7F
S1212DF22B245D263CB62A13BD2B245D2633BD2D255D262DBD2B245D26277A2A1326AA
S1212E10EFB62A1443B72457BD2D255D2615B124572706CE2D897E25D87E2D438139A1
S1152E2F26F97E26D0C10827F97E2CA4535441204120 5B
S9
```

SECURE

```
S00320 0A32
S1212000000044534B0000202A0000000000000000000000000000000000000000092
S121201E0000000000000000000000000000000000000000000000000000000000A0
S121203C0000000000000000000000000000000000000000000000000000000000082
S121205A0000000000000000000000000000000000000000000000000000000000064
S1212078000000000000000000000000000000000000000000000000000000000046
S121209600000000000000000000000000000000CE20006F096F063F2FD609
S12120B425C103262F7D0027260A962881032204A7092014CE20CE3F3139204E554D37
S12120D2424552204552524F520D3F2FD625C13A26E43F2FD625C1012714CE20F23F69
S11220F031392046 4F524D4154204552524F52E0
S12120FF0DBE20FF202A9622B7202C3F2FD625C12F26DA7C202C3F2FD625C10126CF70
S121211DD622FB202CCE20103F05FE202A3F053F34313131315D26B7CE20003F17A638
S121213B05271D81012616CE21483F31392046494C45204E4F5420464F554E440D3F62
S12121591E39EE273F05CE20103F05C60C3F123131313127 07CE20003F1A20C53F2FC3
S1212177962681042703 7E20EC3F2FD625C10327037E20C87D002726F8CE2000EE27CF
S11E21959628C60E3F0AA700CE200063063F136F064D27037E2158392000 6361
S9
```

SET

```
S00320 00DC
S12120003F2FDE20A60091432601 39D625C1012708CE208D3F317E207FDE20EE003F59
S121201E2FD625C13D26EC3F2FD625C10327 42C10126498C4458261ADE20A6008148CA
S121203C260786FF97407E207F814626C8860097407E207F8C5053262 5DE20A600810E
S121205A4E260786FF97427F207F815926A9860097427E207F8D40250796 28A7007F6D
S1212078207FCE209A3F31CE20AB3F313F30DE20DF237E200053594E544158204552FB
S1212096524F520D494E56414C494420534554205041524D0D5345542D20043F02CE6C
S12120B420CCA1002608E1012604FF020C39080808086D0026EC0D3942530039444CCB
S12120D2003A4450003B5744003D4E4C003E5442003F454A0041455300434C440044DF
S11220F04C5700464F524D4154204552524F52C7
S10420FF0DCF
S9
```

STATUS

```
S00320 00DC
S12120003F03FF206EA60026 0139B7206 4B7206AA601B7206 5B7206BA602B7206 6B7AC
S121201F206CA603EA04A1052613E106260F3F05CE20643F313F06C6073F0A20CA3FFC
S121203C05FE206EA1052617E1062613A600B7206AA601B7206BA602B7206C3F0620CE
S119205AD1080808080808080820DC20202020 3D20202020 0D000001D
S9
```

```
0001    0000 0000    N        NAM SWTPDK
0002                  *
0003                  * DISK DRIVERS FOR SOUTHWEST TECHNICAL PRODUCTS
0004                  *
0005                  * COMMANDS
0006                  *
0007    0000 000B     FDRSC  EQU $0B        RESTORE
0008    0000 001B     FDSKI  EQU $1B        SEEK
0009    0000 008C     FDRDC  EQU $8C        READ A SECTOR
0010    0000 00AC     FDWRC  EQU $AC        WRITE A SECTOR
0011                  *
0012    0000 8014     DRVREG EQU $8014
0013    0000 8018     CMDREG EQU $8018
0014    0000 8019     TRKREG EQU $8019
0015    0000 801A     SECREG EQU $801A
0016    0000 801B     DATREG EQU $801B
0017                  *
0018                  * FCB DEFINITIONS:
0019                  *
0020    0000 0005     FCBSTA EQU 5  STATUS
0021    0000 0007     FCBDBA EQU 7  DATA BUFFER ADDRESS
0022    0000 0009     FCBDRV EQU 9  DRIVE#
0023    0000 000A     FCBTRK EQU 10 TRACK#
0024    0000 000B     FCBSCT EQU 11 SECTOR#
0025                  *
0026                  * USER REGISTERS
0027                  *
0028    0000 0006     UA     EQU 6  RETURN "A"
0029    0000 0007     UXH    EQU 7  X-REG (FCBADR)
0030                  *
0031    0000 0000    N        ENT @INTDK
0032    0000 000E    N        ENT .RDSEC
0033    0000 0040    N        ENT .WTSEC
0034                  *
0035    0000 0000     CTRK0  EQU 0
0036    0000 0001     CTRK1  EQU 1
0037    0000 0002     CTRK2  EQU 2
0038    0000 0003     CTRK3  EQU 3
0039    0000 0004     CDRIV  EQU 4
0040    0000 0005     RCNT   EQU 5
0041                  *
0042                  * INIT THE DISK SYSTEM
0043                  *
0044    0000 86 FF    @INTDK LDA A #$FF
0045    0002 97 00           STA A CTRK0
0046    0004 97 01           STA A CTRK1
0047    0006 97 02           STA A CTRK2
0048    0008 97 03           STA A CTRK3
0049    000A 7F 0004         CLR CDRIV
0050    000D 39              RTS
0051                  *
0052                  * READ A SECTOR
0053                  *
0054                  .RDSEC TABX          GET FCBADR
0055 +  000E 3F               SWI
0056 +  000F 03               FCB 3
0057    0010 A6 09           LDA A FCBDRV,X GET DRIVE#
0058    0012 CE 0000         LDX #CTRK0     POINT TO TRACK TABLE
0059                         ADDAX          ADD OFFSET
0060 +  0015 3F               SWI
```

```
0061 +  0016 09                FCB 9
0062    0017 E6 00              LDA B 0,X       GET ENTRY
0063    0019 C1 FF              CMP B #$FF      INITIALIZED?
0064    001B 26 05              BNE RDSEC1      YES
0065                    *
0066    001D BD 0152 R          JSR RESTOR      SEEK TRACK0
0067    0020 25 53              BCS QUIT10      ERROR
0068                    *
0069    0022 BD 0127 R RDSEC1   JSR DRIVE
0070    0025 25 4E              BCS QUIT10      ERROR
0071                    *
0072    0027 86 05              LDA A #5
0073    0029 97 05              STA A RCNT
0074                    *
0075    002B 30         RDSEC2  TSX
0076    002C EE 07              LDX UXH,X       GET FCBADR
0077    002E A6 0A              LDA A FCBTRK,X  GET TRACK#
0078    0030 E6 0B              LDA B FCBSCT,X  GET SECTOR
0079    0032 EE 07              LDX FCBDBA,X    GET BUFFER ADDRESS
0080                    *
0081    0034 BD 007F R          JSR @READ       GET A SECTOR
0082    0037 27 3C              BEQ QUIT10      OK
0083                    *
0084    0039 7A 0005            DEC RCNT        RETRY AGAIN?
0085    003C 26 ED              BNE RDSEC2      YES
0086    003E 20 30              BRA QUIT
0087                    *
0088                    * WRITE A SECTOR
0089                    *
0090                    WTSEC   TABX            X=FCBADR
0091 +  0040 3F                 SWI
0092 +  0041 03                 FCB 3
0093    0042 A6 09              LDA A FCBDRV,X  GET DRIVE#
0094    0044 CE 0000            LDX #CTRK0      POINT TO TABLE
0095                            ADDAX           ADD OFFSET
0096 +  0047 3F                 SWI
0097 +  0048 09                 FCB 9
0098    0049 E6 00              LDA B 0,X       GET ENTRY
0099    004B C1 FF              CMP B #$FF      DRIVE INITIALIZED?
0100    004D 26 05              BNE WTSEC1      YES
0101                    *
0102    004F BD 0152 R          JSR RESTOR
0103    0052 25 21              BCS QUIT10
0104                    *
0105    0054 BD 0127 R WTSEC1   JSR DRIVE
0106    0057 25 1C              BCS QUIT10
0107                    *
0108    0059 86 05              LDA A #5
0109    005B 97 05              STA A RCNT
0110                    *
0111    005D 30         WTSEC2  TSX
0112    005E EE 07              LDX UXH,X       GET FCBADR
0113    0060 A6 0A              LDA A FCBTRK,X  GET TRACK#
0114    0062 E6 0B              LDA B FCBSCT,X  GET SECTOR#
0115    0064 EE 07              LDX FCBDBA,X    GET DATA BUFFER ADDRESS
0116                    *
0117    0066 BD 00AF R          JSR @WRITE
0118    0069 27 0A              BEQ QUIT10      OK
0119                    *
0120    006B 7A 0005            DEC RCNT        TRY AGAIN?
0121    006E 26 ED              BNE WTSEC2      YES
0122                    *
0123                    *
0124    0070 4D         QUIT    TST A           OK?
0125    0071 27 02              BEQ QUIT10      YES
0126                    *
0127    0073 86 05              LDA A #5        ERROR
0128                    *
0129    0075 30         QUIT10  TSX
0130    0076 A7 06              STA A UA,X      STORE RC IN USER A-REG
0131    0078 EE 07              LDX UXH,X       GET FCBADR
0132    007A AA 05              ORA A FCBSTA,X
0133    007C A7 05              STA A FCBSTA,X
0134    007E 39                 RTS
0135                    *
0136                    *
0137                    * READ A SECTOR (A=TRK, B=SECTOR)
0138                    *
0139    007F BD 00D8 R @READ    JSR SEEK
0140    0082 86 8C              LDA A #FDRDC    READ
0141    0084 B7 8018            STA A CMDREG
0142    0087 BD 00F5 R          JSR DEL30U      DELAY
0143    008A C6 80              LDA B #128      128 BYTES/SECTOR
0144                    *
0145    008C B6 8018    READ1   LDA A CMDREG
0146    008F 85 02              BIT A #$02      DATA REG FULL?
0147    0091 26 06              BNE READ2       YES
0148                    *
0149    0093 85 01              BIT A #1        BUSY?
0150    0095 26 F5              BNE READ1       YES
0151                    *
0152    0097 20 0B              BRA READ3       ERROR
0153                    *
0154    0099 B6 801B    READ2   LDA A DATREG    GET A BYTE
0155    009C A7 00              STA A 0,X       STORE IN BUFFER
0156    009E 08                 INX
0157    009F 5A                 DEC B
0158    00A0 26 EA              BNE READ1       DO AGAIN
0159                    *
0160    00A2 8D 03              BSR WBUSY       WAIT TILL DONE
0161                    *
0162    00A4 84 1C      READ3   AND A #$1C      MASK OF STATUS BITS
0163    00A6 39                 RTS
0164                    *
0165    00A7 B6 8018    WBUSY   LDA A CMDREG
0166    00AA 85 01              BIT A #1        BUSY?
0167    00AC 26 F9              BNE WBUSY       YES
0168                    *
0169    00AE 39                 RTS
0170                    *
0171                    * WRITE A SECTOR (A=TRK, B=SECTOR, X=DATA BUFFER)
0172                    *
0173    00AF BD 00D8 R @WRITE   JSR SEEK
0174    00B2 86 AC              LDA A #FDWRC    WRITE COMMAND
0175    00B4 B7 8018            STA A CMDREG
0176    00B7 BD 00F5 R          JSR DEL30U
0177    00BA C6 80              LDA B #128      128 BYTES/SECTOR
0178                    *
0179    00BC B6 8018    WRITE1  LDA A CMDREG
0180    00BF 85 02              BIT A #2        DREG EMPTY?
0181    00C1 26 06              BNE WRITE2      YES
0182                    *
```

```
0183   00C3 85 01          BIT A #1      BUSY?
0184   00C5 26 F5          BNE WRITE1    YES
0185                *
0186   00C7 20 0C          BRA WRITE3    ERROR
0187                *
0188   00C9 A6 00    WRITE2 LDA A 0,X     GET A BYTE
0189   00CB B7 801B        STA A DATREG
0190   00CE 08             INX
0191   00CF 5A             DEC B
0192   00D0 26 EA          BNE WRITE1    DO AGAIN
0193                *
0194   00D2 BD 00A7 R       JSR WBUSY     WAIT FOR BUSY
0195                *
0196   00D5 84 5C    WRITE3 AND A #$5C    MASK OFF STATUS BITS
0197   00D7 39             RTS
0198                *
0199                * SEEK TRACK IN A-REG
0200                *
0201   00D8 B1 8019  SEEK   CMP A TRKREG  ON TRACK?
0202   00DB 27 11          BEQ SEEK2     YES
0203                *
0204   00DD B7 801B        STA A DATREG  NO, STORE TRACK#
0205   00E0 BD 00F5 R      JSR DEL30U
0206   00E3 86 1B          LDA A #FDSKI  SEEK COMMAND
0207   00E5 B7 8018        STA A CMDREG
0208   00E8 BD 00F5 R      JSR DEL30U
0209   00EB BD 00A7 R      JSR WBUSY     WAIT FOR BUSY
0210                *
0211   00EE F7 801A  SEEK2  STA B SECREG        SET SECTOR
0212   00F1 BD 00F5 R      JSR DEL30U
0213   00F4 39             RTS
0214                *
0215                * DELAY 30 USECS
0216                *
0217   00F5 08       DEL30U INX
0218   00F6 09              DEX
0219   00F7 08              INX
0220   00F8 09              DEX
0221   00F9 08              INX
0222   00FA 09              DEX
0223   00FB 08              INX
0224   00FC 09              DEX
0225   00FD 39              RTS
0226                *
0227                * DELAY ONE SECOND
0228                *
0229                DEL1S  PSHX
0230 +  00FE 3F              SWI
0231 +  00FF 05              FCB 5
0232   0100 36              PSH A
0233   0101 86 02           LDA A #2
0234   0103 CE 0000         LDX #0000
0235                *
0236   0106 08       DEL1SA INX
0237   0107 26 FD           BNE DEL1SA
0238                *
0239   0109 4A              DEC A
0240   010A 26 FA           BNE DEL1SA
0241                *
0242   010C 32              PUL A
0243                        PULX
0244 +  010D 3F              SWI
0245 +  010E 06              FCB 6
0246   010F 39             RTS
0247                *
0248                * TURN ON MOTORS (A=UNIT#)
0249                *
0250   0110 F6 8018  MOTOR  LDA B CMDREG
0251   0113 C4 80           AND B #$80    READY?
0252   0115 27 0E           BEQ MOTOR1    YES
0253                *
0254   0117 BD 00FE R       JSR DEL1S     DELAY A SECOND
0255                *
0256   011A F6 8018         LDA B CMDREG
0257   011D C4 80           AND B #$80    READY?
0258   011F 27 04           BEQ MOTOR1    YES
0259                *
0260   0121 86 0A           LDA A #10     ERROR CODE
0261   0123 0D              SEC
0262   0124 39              RTS
0263                *
0264   0125 0C       MOTOR1 CLC
0265   0126 39              RTS
0266                *
0267                * DRIVE SELECT (A=DRIVE#)
0268                *
0269   0127 84 03    DRIVE  AND A #3
0270   0129 36              PSH A
0271   012A 8D 1A           BSR DSEL1
0272   012C 8D E2           BSR MOTOR
0273   012E 25 14           BCS DRIV1
0274                *
0275   0130 F6 8019         LDA B TRKREG  GET CURRENT TRACK
0276   0133 E7 00           STA B 0,X     SAVE IN TABLE
0277   0135 B7 8014         STA A DRVREG  INIT REGISTER
0278   0138 97 04           STA A CDRIV   NEW CURRENT DRIVE
0279   013A 8D 0A           BSR DSEL1
0280   013C A6 00           LDA A 0,X     GET CURRENT TRK
0281   013E B7 8019         STA A TRKREG
0282   0141 BD 00F5 R       JSR DEL30U
0283   0144 32       DRIV1  PUL A
0284   0145 39              RTS
0285                *
0286   0146 CE 0000  DSEL1  LDX #CTRK0    POINT TO TABLE
0287   0149 D6 04           LDA B CDRIV
0288   014B 27 04           BEQ DSEL3
0289                *
0290   014D 08       DSEL2  INX
0291   014E 5A              DEC B
0292   014F 26 FC           BNE DSEL2
0293                *
0294   0151 39       DSEL3  RTS
0295                *
0296                * RESTORE SEEK TRACK0 (X=CTRKx)
0297                *
0298   0152 36       RESTOR PSH A
0299   0153 BD 0127 R       JSR DRIVE
0300   0156 25 0E           BCS RESTR1    ERROR
0301                *
0302   0158 86 0B           LDA A #FDRSC  RESTORE COMMAND
0303   015A B7 8018         STA A CMDREG
0304   015D BD 00F5 R       JSR DEL30U
```

```
0305                        *
0306   0160 BD 00A7 R       RESTR1 JSR WBUSY
0307   0163 6F 00                  CLR 0,X         CTRKx:=00
0308   0165 0C                     CLC
0309                        *
0310   0166 32              RESTR2 PUL A
0311   0167 39                     RTS
0312                        *
0313                        END
```

```
.RDSEC  000E RN
.WTSEC  0040 RN
@INITDK 0000 RN
@READ   007F R
@WRITE  00AF R
ADDABX  2219 M
ADDAX   2232 M
ADDBX   224B M
ADDXAB  2200 M
BASEQU  2A2A M
CDRIV   0004
CHAIN   243A M
CLOSE   2369 M
CMDREG  8018
CMPC    231B M
CMWC    2572 M
CTRK0   0000
CTRK1   0001
CTRK2   0002
CTRK3   0003
DATREG  801B
DEL1S   00FE R
DEL1SA  0106 R
DEL30U  00F5 R
DELETE  2420 M
DIV16   2524 M
DRIV1   0144 R
DRIVE   0127 R
DRVREG  8014
DSEL1   0146 R
DSEL2   014D R
DSEL3   0151 R
FCBDBA  0007
FCBDEF  2650 M
FCBDRV  0009
FCBSCT  000B
FCBSTA  0005
FCBTRK  000A
FDRDC   008C
FDRSC   000B
FDSKI   001B
FDWRC   00AC
FIBDEF  2940 M
FMTFCB  2488 M
FMTS    2558 M
GETDR   23EC M
GTCMD   24F0 M
INDEX   24BC M
INITDK  253E M
IOHDR   2335 M
LOADB   246E M
MOTOR   0110 R
MOTOR1  0125 R
MOVC    2301 M
MOVS    24A2 M
MUL16   22E7 M
MUL8    22CD M
NXTOK   24D6 M
OPEN    234F M
OPEND   239E M
PRTERR  2454 M
PRTMSG  250A M
PSHALL  2151 M
PSHX    21CE M
PULLAL  216A M
PULX    21E7 M
PUTDR   2406 M
QUIT    0070 R
QUIT10  0075 R
RCBDEF  258C M
RCNT    0005
RDSEC1  0022 R
RDSEC2  002B R
READ    23B8 M
READ1   008C R
READ2   0099 R
READ3   00A4 R
RESTOR  0152 R
RESTR1  0166 R
REWIND  2384 M
SECREG  801A
SEEK    00D8 R
SEEK2   00EE R
SUBABX  227F M
SUBAX   2299 M
SUBBX   22B3 M
SUBXAB  2265 M
SWTPDK  0000 RN
TABX    219C M
TRKREG  8019
TXAB    2183 M
UA      0006
UXH     0007
WBUSY   00A7 R
WRITE   23D2 M
WRITE1  00BC R
WRITE2  00C9 R
WRITE3  00D5 R
WTSEC1  0054 R
WTSEC2  005D R
XABX    21B5 M
```

```
0001   0000 0000    N       NAM BOOT
0002                *
0003                * SWTPC CP/68 BOOTSTRAP PROGRAM
0004                * ASSUMES SYSTEM FILE LINKED AS FOLLOWS:
0005                *
0006                *      TRACK 0, SECTOR 1, BYTE 122-FIRST TRACK
0007                *                             123-FIRST SECTOR
0008                *                             124-LAST TRACK
0009                *                             125-LAST SECTOR
0010                *                             126,7 FREE-SPACE HEADER
0011                *
0012                * BOOTS SYSTEM FROM DRIVE 0:
0013                *
0014                * DEFINE DISK-DRIVE INTERFACE ADDRESSING
0015                *
0016   0000 000B    FDRSC  EQU $0B       RESTORE
0017   0000 001B    FDSKI  EQU $1B       SEEK
0018   0000 008C    FDRDC  EQU $8C       READ A SECTOR
0019                *
0020   0000 8014    DRVREG EQU $8014     DRIVE REGISTER
0021   0000 8018    CMDREG EQU $8018     COMMAND REGISTER
0022   0000 8019    TRKREG EQU $8019     TRACK REGISTER
0023   0000 801A    SECREG EQU $801A     SECTOR REGISTER
0024   0000 801B    DATREG EQU $801B     DATA REGISTER
0025                *
0026                * NOTE: ALL VARIABLES IN COMMON, CODE IS ROM-ABLE
0027                *
0028   0000 0000  C         CMN STACK,16
0029   0000 0010  C         CMN BUFFER,128
0030   0000 0090  C         CMN FTS,2
0031   0000 0092  C         CMN LTS,2
0032   0000 0094  C         CMN PTS,2
0033   0000 0096  C         CMN INDEX,2
0034   0000 0098  C         CMN SAVEX,2
0035   0000 009A  C         CMN SAVEX2,2
0036   0000 009C  C         CMN ADDRES,2
0037   0000 009E  C         CMN FCNT,1
0038   0000 009F  C         CMN RCNT,1
0039                *
0040                * ERROR JUMP VECTOR
0041                *
0042   0000 7E E113 ERROR   JMP $E113
0043                *
0044                * BEGIN BOOT HERE
0045                *
0046   0003 8E 000F C START  LDS #STACK+15 INIT. STACK POINTER
0047   0006 86 01           LDA A #1      SECTOR=1
0048   0008 C6 01           LDA B #1      TRACK=1
0049   000A BD 013F R       JSR DRIVE     SETUP DRIVE 0
0050   000D 86 0B           LDA A #FDRSC  ISSUE RESTORE COMMAND
0051   000F B7 8018         STA A CMDREG
0052   0012 BD 0136 R       JSR DEL30U    30 USEC DELAY
0053   0015 B6 8018 START2  LDA A CMDREG
0054   0018 85 01           BIT A #1      BUSY?
0055   001A 26 F9           BNE START2    YES
0056                *
0057                * NOW GET SYSTEM LINK INFORMATION
0058                *
0059   001C 86 01           LDA A #1
0060   001E C6 00           LDA B #0      TRACK 0, SECTOR 1
0061   0020 CE 0010 C       LDX #BUFFER
0062   0023 BD 00CB R       JSR RDSEC     READ LINK SECTOR
0063   0026 CE 0010 C       LDX #BUFFER
0064   0029 A6 7A           LDA A 122,X   GET FIRST T/S
0065   002B E6 7B           LDA B 123,X
0066   002D B7 0090 C       STA A FTS
0067   0030 F7 0091 C       STA B FTS+1
0068   0033 A6 7C           LDA A 124,X   GET LAST T/S
0069   0035 E6 7D           LDA B 125,X
0070   0037 B7 0092 C       STA A LTS
0071   003A F7 0093 C       STA B LTS+1
0072   003D CE 0014 C       LDX #BUFFER+4
0073   0040 FF 0096 C       STX INDEX     INIT. BUFFER INDEX
0074   0043 B6 0091 C       LDA A FTS+1
0075   0046 F6 0090 C       LDA B FTS
0076   0049 B7 0095 C       STA A PTS+1   INIT. PRESENT T/S
0077   004C F7 0094 C       STA B PTS
0078   004F CE 0010 C       LDX #BUFFER
0079   0052 BD 00CB R       JSR RDSEC     READ FIRST SECTOR
0080                *
0081                * NOW LOAD SYSTEM FILE INTO MEMORY
0082                *
0083   0055 8D 3A   BOOT1   BSR GETBYT    GET A DATA BYTE FROM FILE
0084   0057 81 16           CMP A ##$16   TRANSFER-ADDRESS?
0085   0059 26 0C           BNE BOOT2     NO
0086                *
0087   005B 8D 34           BSR GETBYT
0088   005D B7 009C C       STA A ADDRES  GET TRANSFER ADDRESS
0089   0060 8D 2F           BSR GETBYT
0090   0062 B7 009D C       STA A ADDRES+1
0091   0065 20 EE           BRA BOOT1     GET NEW DATA FRAME
0092                *
0093   0067 81 02   BOOT2   CMP A ##$02   DATA FRAME?
0094   0069 26 21           BNE BOOT4     NO
0095                *
0096   006B 8D 24           BSR GETBYT
0097   006D B7 0098 C       STA A SAVEX   GET ADDRESS
0098   0070 8D 1F           BSR GETBYT
0099   0072 B7 0099 C       STA A SAVEX+1
0100   0075 8D 1A           BSR GETBYT
0101   0077 B7 009E C       STA A FCNT    GET FRAME COUNTER
0102                *
0103   007A 8D 15   BOOT3   BSR GETBYT    GET DATA BYTE
0104   007C FE 0098 C       LDX SAVEX
0105   007F A7 00           STA A 0,X     STORE BYTE
0106   0081 08              INX
0107   0082 FF 0098 C       STX SAVEX
0108   0085 7A 009E C       DEC FCNT      COUNT DOWN
0109   0088 26 F0           BNE BOOT3
0110                *
0111   008A 20 C9           BRA BOOT1     GET NEW DATA FRAME
0112                *
0113   008C FE 009C C BOOT4  LDX ADDRES    GET TRANSFER ADDRESS
0114   008F 6E 00           JMP 0,X       GO THERE
0115                *
0117                * READ A DATA BYTE FROM SYSTEM FILE
0118                * RETURN BYTE IN 'A' REGISTER
0119                *
0120   0091 FE 0096 C GETBYT LDX INDEX
0121   0094 8C 0090 C       CPX #BUFFER+128      NEED NEW SECTOR?
0122   0097 27 07           BEQ GETSEC           YES
```

```
0123                        *
0124   0099 A6 00           GETBYT  LDA A 0,X          GET BYTE
0125   009B 08                      INX
0126   009C FF 0096 C               STX INDEX         MOVE POINTER
0127   009F 39                      RTS
0128                        *
0129   00A0 F6 0094 C       GETSEC  LDA B PTS
0130   00A3 B6 0095 C               LDA A PTS+1       CHECK FOR LAST SECTOR
0131   00A6 B1 0093 C               CMP A LTS+1
0132   00A9 26 07                   BNE GETS2         NOT LAST
0133                        *
0134   00AB F1 0092 C               CMP B LTS
0135   00AE 26 02                   BNE GETS2         NOT LAST
0136                        *
0137   00B0 20 DA                   BRA BOOT4         EOF-GO TO TRANSFER ADDRESS
0138                        *
0139   00B2 CE 0010 C       GETS2   LDX #BUFFER
0140   00B5 E6 00                   LDA B 0,X         GET FORWARD T/S LINK
0141   00B7 A6 01                   LDA A 1,X
0142   00B9 F7 0094 C               STA B PTS         UPDATE PRESENT T/S
0143   00BC B7 0095 C               STA A PTS+1
0144   00BF 8D 0A                   BSR RDSEC         READ NEW SECTOR
0145   00C1 CE 0014 C               LDX #BUFFER+4
0146   00C4 A6 00                   LDA A 0,X         GET DATA BYTE
0147   00C6 08                      INX
0148   00C7 FF 0096 C               STX INDEX         RE-INIT. INDEX
0149   00CA 39                      RTS
0150                        *
0152                        * SINGLE-SECTOR READ ROUTINE
0153                        *
0154                        *       DRIVE=0
0155                        *       TRACK='B'
0156                        *       SECTOR='A'
0157                        *       BUFFER='X'
0158                        *
0159   00CB 36              RDSEC   PSH A             SAVE SECTOR
0160   00CC 37                      PSH B             SAVE TRACK
0161   00CD FF 009A C               STX SAVEX2        SAVE BUFFER POINTER
0162   00D0 BD 013F R               JSR DRIVE         SETUP DRIVE 0
0163   00D3 FE 009A C               LDX SAVEX2        RESTORE BUFFER POINTER
0164   00D6 86 05                   LDA A #5          FIVE RETRIES
0165   00D8 B7 009F C               STA A RCNT
0166                        *
0167   00DB 33              RDSEC2  PUL B             GET TRACK
0168   00DC 32                      PUL A             GET SECTOR
0169   00DD FE 009A C               LDX SAVEX2        GET BUFFER POINTER
0170   00E0 36                      PSH A
0171   00E1 37                      PSH B             PUT BACK ONTO STACK
0172   00E2 BD 011B R       RDSEC3  JSR SEEK
0173   00E5 86 8C                   LDA A #FDRDC      READ COMMAND
0174   00E7 B7 8018                 STA A CMDREG
0175   00EA BD 0136 R               JSR DEL30U        30 USEC WAIT
0176   00ED C6 80                   LDA B #128        SECTOR SIZE=128 BYTES
0177   00EF B6 8018         READ1   LDA A CMDREG
0178   00F2 85 02                   BIT A #$02        REG. FULL?
0179   00F4 26 06                   BNE READ2         YES
0180                        *
0181   00F6 85 01                   BIT A #$01        BUSY?
0182   00F8 26 F5                   BNE READ1         YES
0183                        *
0184   00FA 20 09                   BRA READ3         ERROR
0185                        *
0186   00FC B6 801B         READ2   LDA A DATREG      GET A BYTE
0187   00FF A7 00                   STA A 0,X         SAVE IT
0188   0101 08                      INX
0189   0102 5A                      DEC B             COUNT DOWN
0190   0103 26 EA                   BNE READ1         DO AGAIN
0191                        *
0192   0105 B6 8018         READ3   LDA A CMDREG
0193   0108 85 01                   BIT A #$01        BUSY?
0194   010A 26 F9                   BNE READ3         YES
0195                        *
0196   010C 84 1C           READ4   AND A #$1C        MASK OFF STATUS BITS
0197   010E 27 08                   BEQ READ5         GOOD?
0198                        *
0199   0110 7A 009F C               DEC RCNT          NO, TRY AGAIN?
0200   0113 26 C6                   BNE RDSEC2
0201                        *
0202   0115 7E 0000 R       QUIT    JMP ERROR         FAILED 5 TIMES, ERROR
0203                        *
0204   0118 33              READ5   PUL B             CLEAN STACK
0205   0119 32                      PUL A
0206   011A 39                      RTS               GOOD READ
0207                        *
0208                        * SEEK TRACK IN "B" , SECTOR IN "A"
0209                        *
0210   011B F1 8019         SEEK    CMP B TRKREG      ON TRACK?
0211   011E 27 13                   BEQ SEEK2         YES
0212                        *
0213   0120 F7 801B                 STA B DATREG      NO
0214   0123 8D 11                   BSR DEL30U        30 USEC WAIT
0215   0125 C6 1B                   LDA B #FDSKI      SEEK COMMAND
0216   0127 F7 8018                 STA B CMDREG
0217   012A 8D 0A                   BSR DEL30U        30 USEC WAIT
0218   012C F6 8018         SEEK1   LDA B CMDREG
0219   012F C5 01                   BIT B #$01        BUSY?
0220   0131 26 F9                   BNE SEEK1         WAIT UNTIL DONE
0221                        *
0222   0133 B7 801A         SEEK2   STA A SECREG      SAVE SECTOR
0223                        *
0224                        * 30 USEC DELAY FOR CONTROLLER CHIP
0225                        *
0226   0136 08              DEL30U  INX
0227   0137 09                      DEX
0228   0138 08                      INX
0229   0139 09                      DEX
0230   013A 08                      INX
0231   013B 09                      DEX
0232   013C 08                      INX
0233   013D 09                      DEX
0234   013E 39                      RTS
0235                        *
0236                        *
0237                        * SELECT DRIVE 0
0238                        *
0239   013F F6 8018         DRIVE   LDA B CMDREG
0240   0142 C4 80                   AND B #$80        READY?
0241   0144 27 16                   BEQ DRIVE1        YES
0242                        *
0243   0146 CE 0000                 LDX #0000         1 SECOND DELAY
0244   0149 08              DRV0    INX
0245   014A 26 FD                   BNE DRV0
```

```
0246                        *
0247   014C CE 0000         DRV0   LDX #0000
0248   014F 08              DRV1   INX
0249   0150 26 FD                  BNE DRV1
0250                        *
0251   0152 F6 8018                 LDA B CMDREG
0252   0155 C4 80                   AND B #$80     READY?
0253   0157 27 03                   BEQ DRIVE1     YES
0254                        *
0255   0159 7E 0000 R       DRIVE0 JMP ERROR       NO, ERROR
0256                        *
0257   015C 7F 8014         DRIVE1 CLR DRVREG      DRIVE=0
0258   015F 20 D5                   BRA DEL30U     30 USEC WAIT
0259                        *
0260                               END
```

```
ADDRES 009C C
BOOT   0000 RN
BOOT1  0055 R
BOOT2  0067 R
BOOT3  007A R
BOOT4  008C R
BUFFER 0010 C
CMDREG 8018
DATREG 801B
DEL30U 0136 R
DRIVE  013F R
DRIVE0 0159 R
DRIVE1 015C R
DRV0   0149 R
DRV1   014F R
DRVREG 8014
ERROR  0000 R
FCNT   009E C
FDRDC  008C
FDRSC  000B
FDSKI  001B
FTS    0090 C
GETBYT 0091 R
GETS2  00B2 R
GETSEC 00A0 R
INDEX  0096 C
LTS    0092 C
PTS    0094 C
QUIT   0115 R
RCNT   009F C
RDSEC  00CB R
RDSEC2 00DB R
RDSEC3 00E2 R
READ1  00EF R
READ2  00FC R
READ3  0105 R
READ4  010C R
READ5  0118 R
SAVEX  0098 C
SAVEX2 009A C
SECREG 801A
SEEK   011B R
SEEK1  012C R
SEEK2  0133 R
STACK  0000 C
START  0003 R
START2 0015 R
TRKREG 8019
```

```
0001    0000 0000    N        NAM INITER
0002                  *
0003                  * INITIALIZE A DISK FOR CP-68 OPERATING SYSTEM
0004                  *
0005                  * FOR SWTPC 5 INCH FLOPPY DISKS
0006                  *
0007                  * TRACK 0, SECTOR 1          BOOTSTRAP
0008                  * TRACK 0, SECTOR 1          HEADER OF FREE-SPACE LIST
0009                  * TRACK 0, SECTORS 2-18      DIRECTORY SPACE
0010                  * TRACKS 1-35                FREE-SPACE
0011                  *
0012                  * DISK ATTRIBUTES
0013                  *
0014    0000 0080     SECSIZ EQU 128        128 BYTES PER SECTOR
0015    0000 0012     TRKSIZ EQU 18         18 SECTORS PER TRACK
0016    0000 0022     DSKSIZ EQU 34         34 TRACKS ON DISK (LESS TRACK 0)
0017                  *
0018                  * FILE-CONTROL BLOCK ADDRESSES
0019                  *
0020                         FCBDEF
0021 +  0000 0000      FCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0022 +  0000 0002      FCBGDT EQU 2          GENERIC DEVICE TYPE
0023 +  0000 0005      FCBSTA EQU 5          STATUS
0024 +  0000 0006      FCBDTT EQU 6          DATA TRANSFER TYPE
0025 +  0000 0007      FCBDBA EQU 7          DATA BUFFER ADDRESS
0026 +  0000 0009      FCBDRV EQU 9          DRIVE NUMBER
0027 +  0000 000A      FCBTRK EQU 10         TRACK NUMBER
0028 +  0000 000B      FCBSCT EQU 11         SECTOR NUMBER
0029 +  0000 000C      FCBFWD EQU 12         FWD LINK TRACK/SECTOR
0030 +  0000 000E      FCBBAK EQU 14         BACK LINK TRACK/SECTOR
0031 +  0000 0010      FCBNAM EQU 16         FILE NAME (8.3+EOT=13))
0032 +  0000 001D      FCBTYP EQU 29         FILE TYPE
0033 +  0000 001E      FCBACS EQU 30         FILE ACCESS CODE
0034 +  0000 001F      FCBFTS EQU 31         FIRST TRACK/SECTOR
0035 +  0000 0021      FCBLTS EQU 33         LAST TRACK/SECTOR
0036 +  0000 0023      FCBNMS EQU 35         NUMBER OF SECTORS
0037 +  0000 0025      FCBNFB EQU 37         NEXT FCB IN ACTIVE CHAIN
0038 +  0000 0027      FCBIND EQU 39         INDEX INTO DATA BUFFER
0039 +  0000 0029      FCBSCF EQU 41         SPACE COMPRESSION FLAG
0040                  *
0041    0000 0002     FCBSPC RMB 2          FILE-CONTROL BLOCK
0042    0002 44              FCC 'DSK'      DISK
0043    0005 0001            RMB 1
0044    0006 FF              FCB $FF        OUTPUT
0045    0007 0023            RMB 35
0046                  *
0047    002A 0080     BUFFER RMB SECSIZ     SECTOR BUFFER
0048                  *
0049                  * COMMAND-LINE INTERPRETER BASE-PAGE LOCATIONS
0050                  *
0051                         BASEQU
0052 +  00AA 0020      DESCRA EQU $20  DESCRIPTOR ADDRESS(2)
0053 +  00AA 0022      DESCRC EQU $22  DESCRIPTOR COUNT
0054 +  00AA 0023      CUCHAR EQU $23  CURRENT CHAR (2)
0055 +  00AA 0025      RC     EQU $25  TOKEN RETURN CODE
0056 +  00AA 0026      CLASS  EQU $26  TOKEN CLASS
0057 +  00AA 0027      VALUE  EQU $27  BIN VALUE/TRANSFER ADDRESS (2)
0058 +  00AA 0029      FCBCHN EQU $29  TOP OF FCB CHAIN (2)
0059 +  00AA 002B      FRETAB EQU $2B  DISK FREE SPACE POINTER (8)
0060 +  00AA 0033      BMEM   EQU $33  START OF TRANSIENT AREA(2)
0061 +  00AA 0035      EMEM   EQU $35  END OF TRANSIENT AREA (2)
0062 +  00AA 0037      CMEM   EQU $37  NEXT AVAIL TRANSIENT AREA (2)
0063 +  00AA 0039      BS     EQU $39  BACKSPACE CHAR
0064 +  00AA 003A      DL     EQU $3A  DELETE LINE CHAR
0065 +  00AA 003B      DP     EQU $3B  DEPTH; LINES/PAGE
0066 +  00AA 003C      DPCNT  EQU $3C  DEPTH TEMP
0067 +  00AA 003D      WD     EQU $3D  WIDTH; CHARS/LINE
0068 +  00AA 003E      NL     EQU $3E  NULL COUNT
0069 +  00AA 003F      TB     EQU $3F  TAB CHAR
0070 +  00AA 0040      DX     EQU $40  DUPLEX; FF=H, 00=F
0071 +  00AA 0041      EJ     EQU $41  EJECT COUNT
0072 +  00AA 0042      PS     EQU $42  PAUSE; 00=YES
0073 +  00AA 0043      ES     EQU $43  ESCAPE CHAR
0074 +  00AA 0044      LDP    EQU $44  DEPTH LINES/PAGE
0075 +  00AA 0045      LDPCNT EQU $45  DEPTH TEMP
0076 +  00AA 0046      LWD    EQU $46  WIDTH CHARS/LINE
0077                  *
0078    00AA 49       PROMPT FCC 'INIT. DISK IN DRIVE '
0079    00BE 0001     DRVNO  RMB 1
0080    00BF 20              FCC ' ? '
0081    00C2 04              FCB $04
0082                  *
0083    00C3 00C3    N       ENT .INITR     ENTRY POINT FROM CLI
0084                  *
0085    00C3 96 28    INITR  LDA A VALUE+1  GET DRIVE NUMBER
0086    00C5 84 03           AND A #$03     LIMIT RANGE (SWTPC PERMITS 4 DRIVES)
0087    00C7 CE 0000 R       LDX #FCBSPC    POINT TO FCB
0088    00CA A7 09           STA A FCBDRV,X
0089    00CC 8B 30           ADD A #$30     MAKE DRIVE NUMBER ASCII
0090    00CE B7 00BE R       STA A DRVNO    PUT IN PROMPT LINE
0091    00D1 CE 00AA R       LDX #PROMPT
0092                         PRTMSG         OUTPUT PROMPT
0093 +  00D4 3F               SWI
0094 +  00D5 31               FCB 49
0095                         GTCMD          GET USER RESPONSE
0096 +  00D6 3F               SWI
0097 +  00D7 30               FCB 48
0098    00D8 DE 20           LDX DESCRA
0099    00DA A6 00           LDA A 0,X      GET FIRST CHAR. OF RESPONSE
0100    00DC 81 59           CMP A #'Y      WAS IT 'YES'?
0101    00DE 27 01           BEQ INITR2     IF SO, CONTINUE
0102                  *
0103    00E0 39              RTS            IF NOT, QUIT
0104                  *
0105    00E1 CE 0000 R INITR2 LDX #FCBSPC    POINT TO FCB
0106    00E4 6F 0A           CLR FCBTRK,X   TRACK=0
0107    00E6 86 01           LDA A #1
0108    00E8 A7 0B           STA A FCBSCT,X  SECTOR=1
0109                  *
0110                  * INITIALIZE HEAD OF FREE-SPACE BLOCK
0111                  *
0112                  * ALL ZERO EXCEPT FOR LAST TWO BYTES=TRACK 1, SECTOR 1
0113                  *
0114                         TXAB
0115 +  00EA 3F               SWI
0116 +  00EB 02               FCB 2
0117    00EC CE 002A R       LDX #BUFFER
0118                         XABX
0119 +  00EF 3F               SWI
0120 +  00F0 04               FCB 4
0121    00F1 A7 07           STA A FCBDBA,X
```

```
0122   00F3 E7 08           STA B FCBDBA+1,X
0123                        PSHX
0124 +  00F5 3F              SWI
0125 +  00F6 05              FCB 5
0126                *
0127                * CLEAR OUT BUFFER EXCEPT FOR LAST 2 BYTES
0128                *
0129   00F7 CE 002A R       LDX #BUFFER
0130   00FA C6 7E           LDA B #SECSIZ-2
0131   00FC 4F              CLR A
0132   00FD A7 00     INITR3 STA A 0,X
0133   00FF 08              INX
0134   0100 5A              DEC B
0135   0101 26 FA           BNE INITR3
0136                *
0137   0103 86 01           LDA A #1        TRACK, SECTOR=1
0138   0105 A7 00           STA A 0,X
0139   0107 A7 01           STA A 1,X
0140                        PULX
0141 +  0109 3F              SWI
0142 +  010A 06              FCB 6
0143   010B 8D 7E           BSR WRTBLK      WRITE BLOCK 3
0144   010D 6D 05           TST FCBSTA,X    CHECK FOR DISK ERROR
0145   010F 27 04           BEQ *+6         OK
0146                *
0147   0111 20 4F           BRA INITQ       FATAL DISK ERROR, QUIT
0148                *
0149   0113 20 76     @WRTBL BRA WRTBLK     OUT OF RANGE "BSR WRTBLK"
0150   0115 6C 0B           INC FCBSCT,X    SECTOR=4
0151   0117 7F 00A8 R       CLR BUFFER+SECSIZ-2
0152   011A 7F 00A9 R       CLR BUFFER+SECSIZ-1
0153                *
0154                * INITIALIZE DIRECTORY TO ZERO
0155                *
0156   011D 8D 6C     INITR4 BSR WRTBLK     WRITE DIRECTORY BLOCK
0157   011F 6D 05           TST FCBSTA,X    CHECK FOR DISK ERROR
0158   0121 27 02           BEQ *+4         OK
0159                *
0160   0123 20 3D           BRA INITQ       FATAL DISK ERROR, QUIT
0161                *
0162   0125 A6 0B           LDA A FCBSCT,X
0163   0127 4C              INC A           NEXT SECTOR
0164   0128 81 12           CMP A #TRKSIZ DONE WITH TRACK?
0165   012A 27 04           BEQ INITR5      YES
0166                *
0167   012C A7 0B           STA A FCBSCT,X
0168   012E 20 ED           BRA INITR4      NO, CONTINUE WRITING
0169                *
0170   0130 86 01     INITR5 LDA A #1
0171   0132 A7 0B           STA A FCBSCT,X  SECTOR=1
0172   0134 A7 0A           STA A FCBTRK,X  TRACK=1
0173   0136 16              TAB
0174                *
0175                * INITIALIZE REST OF DISK (FREE-SPACE)
0176                *
0177                *       X=FCB ADDRESS
0178                *       A=TRACK NUMBER
0179                *       B=SECTOR NUMBER
0180                *
0181   0137 5C        INITR6 INC B           MAKE SECTOR LINKAGE
0182   0138 C1 13           CMP B #TRKSIZ+1 END OF TRACK?
0183   013A 26 09           BNE INITR7      NO
0184                *
0185   013C C6 01           LDA B #1        YES, SECTOR=1
0186   013E 4C              INC A           NEXT TRACK
0187   013F 81 23           CMP A #DSKSIZ+1 END OF DISK?
0188   0141 26 02           BNE INITR7      NO
0189                *
0190   0143 4F              CLR A           LAST SECTOR POINTS TO 0,0
0191   0144 5F              CLR B
0192                *
0193   0145 B7 002A R INITR7 STA A BUFFER    TRACK LINK
0194   0148 37              PSH B           SAVE LSEC
0195   0149 8D 33           BSR GETSC       GET PSEC
0196   014B F7 002B R       STA B BUFFER+1 SECTOR LINK
0197   014E 33              PUL B           RESTORE LSEC
0198   014F 8D 3A           BSR WRTBLK      WRITE SECTOR
0199   0151 4D              TST A           DONE? (=0)
0200   0152 26 04           BNE INITR8      NO
0201                *
0202   0154 5D              TST B           DONE? (=0)
0203   0155 26 01           BNE INITR8      NO
0204                *
0205   0157 39              RTS             YES, DONE!!!
0206                *
0207   0158 A7 0A     INITR8 STA A FCBTRK,X
0208   015A 37              PSH B           SAVE LSEC
0209   015B 8D 21           BSR GETSC       GET PSEC
0210   015D E7 0B           STA B FCBSCT,X
0211   015F 33              PUL B           GET LSEC
0212   0160 20 D5           BRA INITR6      KEEP WRITING
0213                *
0214                *
0215                * FATAL ERROR MESSAGE
0216                *
0217   0162 CE 0168 R INITQ  LDX #QMSG       OUTPUT ERROR MESSAGE
0218                        PRTMSG
0219 +  0165 3F              SWI
0220 +  0166 31              FCB 49
0221   0167 39              RTS             RETURN TO CLI
0222                *
0223   0168 49        QMSG   FCC 'INITIALIZATION FAILED'
0224   017D 0D              FCB $0D
0225                *
0226                * CONVERT LSEC TO PSEC
0227                * LSEC IN B-REG
0228                *
0229                GETSC  PSHX            SAVE X-REGISTER
0230 +  017E 3F              SWI
0231 +  017F 05              FCB 5
0232   0180 CE 01FF R       LDX #TBL        POINT TO LOGICAL/PHYSICAL TABLE
0233                        ADDBX           ADD LOGICAL OFFSET
0234 +  0183 3F              SWI
0235 +  0184 0A              FCB 10
0236   0185 09              DEX             SECTOR STARTS AT 1
0237   0186 E6 00           LDA B 0,X       GET PSEC
0238                        PULX            RESTORE X-REG
0239 +  0188 3F              SWI
0240 +  0189 06              FCB 6
0241   018A 39              RTS
0242                *
0243                * WRITE A SECTOR WITH ERROR CHECKING
```

```
                        *
018B 36         WRTBLK  PSH A           SAVE 'A'
018C 4F                 CLR A
018D 6F 05              CLR FCBSTA,X    CLEAR ERROR FLAG
                        IOHDR           ISSUE I/O REQUEST
+  018F 3F               SWI
+  0190 13               FCB 19
0191 A7 05              STA A FCBSTA,X
0193 4D                 TST A           ERROR?
0194 26 02              BNE WRTERR      YES
                        *
0196 32                 PUL A           RESTORE 'A'
0197 39                 RTS
                        *
0198 16         WRTERR  TAB
0199 8D 54              BSR OUTHL       CONVERT LEFT DIGIT
019B B7 01D5 R          STA A ERTYPE
019E 17                 TBA
019F 8D 52              BSR OUTHR       CONVERT RIGHT DIGIT
01A1 B7 01D6 R          STA A ERTYPE+1
                        PSHX            SAVE X
+  01A4 3F               SWI
+  01A5 05               FCB 5
01A6 A6 0B              LDA A FCBSCT,X
01A8 8D 45              BSR OUTHL       MAKE SECTOR NO.  HEX
01AA B7 01E2 R          STA A SECT
01AD A6 0B              LDA A FCBSCT,X
01AF 8D 42              BSR OUTHR
01B1 B7 01E3 R          STA A SECT+1
01B4 A6 0A              LDA A FCBTRK,X
01B6 8D 37              BSR OUTHL       MAKE TRACK NO.  HEX
01B8 B7 01EC R          STA A TRACK
01BB A6 0A              LDA A FCBTRK,X
01BD 8D 34              BSR OUTHR
01BF B7 01ED R          STA A TRACK+1
01C2 CE 01CA R          LDX #DERROR
                        PRTMSG          PRINT ERROR MESSAGE
+  01C5 3F               SWI
+  01C6 31               FCB 49
01C7 3F                 SWI             CALL CP/68
01C8 1F                 FCB 31          "WARMSTART"
01C9 39                 RTS
                        *
01CA 44         DERROR  FCC 'DISK ERROR:'
01D5 0002       ERTYPE  RMB 2
01D7 20                 FCC ' AT SECTOR '
01E2 0002       SECT    RMB 2
01E4 2C                 FCC ', TRACK '
01EC 0002       TRACK   RMB 2
01EE 0D                 FCB $0D
                        *
                        * CONVERT BINARY TO HEX-ASCII HERE
                        *
01EF 44         OUTHL   LSR A           SHIFT RIGHT
01F0 44                 LSR A
01F1 44                 LSR A
01F2 44                 LSR A
                        *
01F3 84 0F      OUTHR   AND A #$0F      GET NIBBLE
01F5 8B 30              ADD A #$30      MAKE ASCII
01F7 81 39              CMP A #$39      >9?
01F9 23 02              BLS *+4         NO
                        *
01FB 8B 07              ADD A #$7       YES
                        *
01FD 39                 RTS
                        *
                        *
                        * LOGICAL/PHYSICAL SECTOR TABLE
                        *
01FE 00                 FCB 00
                        *
01FF 01         TBL     FCB $1
0200 06                 FCB $6
0201 0B                 FCB $B
0202 10                 FCB $10
0203 03                 FCB $3
0204 08                 FCB $8
0205 0D                 FCB $D
0206 12                 FCB $12
0207 05                 FCB $5
0208 0A                 FCB $A
0209 0F                 FCB $F
020A 02                 FCB $2
020B 07                 FCB $7
020C 0C                 FCB $C
020D 11                 FCB $11
020E 04                 FCB $4
020F 09                 FCB $9
0210 0E                 FCB $E
                        *
                        *
                        *
0211 0211     R BOOT    EQU *           BOOT PROGRAM STARTS HERE
                        *
                        *
                        END
```

```
. INITR 00C3 RN
@WRTBL 0113 R
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
BMEM   0033
BOOT   0211 R
BS     0039
BUFFER 002A R
CHAIN  243A M
CLASS  0026
CLOSE  2369 M
CMEM   0037
CMPC   231B M
CMWC   2572 M
CUCHAR 0023
DELETE 2420 M
DERROR 01CA R
DESCRA 0020
DESCRC 0022
DIV16  2524 M
DL     003A
DP     003B
DPCNT  003C
DRVNO  00BE R
DSKSIZ 0022
DX     0040
EJ     0041
EMEM   0035
ENTYPE 01D5 R
ES     0043
FCBACS 001E
FCBBAK 000E
FCBCHN 0029
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBDTT 0006
FCBEQT 0000
FCBFTS 001F
FCBFWD 000C
FCBGDT 0002
FCBIND 0027
FCBLTS 0021
FCBNAM 0010
FCBNFB 0025
FCBNMS 0023
FCBSCF 0029
FCBSCT 000B
FCBSPC 0000 R
FCBSTA 0005
FCBTRK 000A
FCBTYP 001D
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
FRETAB 002B
GETDR  23EC M
GETSC  017E R
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
INITER 0000 RN
INITQ  0162 R
INITR2 00E1 R
INITR3 00FD R
INITR4 011D R
INITR5 0130 R
INITR6 0137 R
INITR7 0145 R
INITR8 0158 R
IOHDR  2335 M
LDP    0044
LDPCNT 0045
LOADB  246E M
LWD    0046
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NL     003E
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
OUTHL  01EF R
OUTHR  01F3 R
PROMPT 00AA R
PRTERR 2454 M
PRTMSG 250A M
PS     0042
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
QMSG   0168 R
RC     0025
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
SECSIZ 0080
SECT   01E2 R
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
TABX   219C M
TB     003F
TBL    01FF R
TRACK  01EC R
TRKSIZ 0012
TXAB   2183 M
VALUE  0027
WD     003D
WRITE  23D2 M
WRTBLK 018B R
WRTERR 0198 R
XABX   21B5 M
```

```
0001    0000 0000    N        NAM FORMATTER
0002                          *
0003                          * PROGRAM TO FORMAT SOFT-SECTORED MINIFLOPPY DISKS
0004                          * ASSUMES SWTPC HARDWARE, W.D. 1771 CONTROLLER
0005                          * FOR CP/68 SYSTEM---35 TRACKS, 18 SECTORS/TRACK
0006                          *
0007                          * COPYRIGHT: 1979...HEMENWAY ASSOCIATES, BOSTON MASS.
0008                          *
0009                                BASEQU        ESTABLISH CP/68 BASEPAGE
0010 +  0000 0020              DESCRA EQU $20  DESCRIPTOR ADDRESS(2)
0011 +  0000 0022              DESCRC EQU $22  DESCRIPTOR COUNT
0012 +  0000 0023              CUCHAR EQU $23  CURRENT CHAR (2)
0013 +  0000 0025              RC     EQU $25  TOKEN RETURN CODE
0014 +  0000 0026              CLASS  EQU $26  TOKEN CLASS
0015 +  0000 0027              VALUE  EQU $27  BIN VALUE/TRANSFER ADDRESS (2)
0016 +  0000 0029              FCBCHN EQU $29  TOP OF FCB CHAIN (2)
0017 +  0000 002B              FRETAB EQU $2B  DISK FREE SPACE POINTER (8)
0018 +  0000 0033              BMEM   EQU $33  START OF TRANSIENT AREA(2)
0019 +  0000 0035              EMEM   EQU $35  END OF TRANSIENT AREA (2)
0020 +  0000 0037              CMEM   EQU $37  NEXT AVAIL TRANSIENT AREA (2)
0021 +  0000 0039              BS     EQU $39  BACKSPACE CHAR
0022 +  0000 003A              DL     EQU $3A  DELETE LINE CHAR
0023 +  0000 003B              DP     EQU $3B  DEPTH; LINES/PAGE
0024 +  0000 003C              DPCNT  EQU $3C  DEPTH TEMP
0025 +  0000 003D              WD     EQU $3D  WIDTH; CHARS/LINE
0026 +  0000 003E              NL     EQU $3E  NULL COUNT
0027 +  0000 003F              TB     EQU $3F  TAB CHAR
0028 +  0000 0040              DX     EQU $40  DUPLEX; FF=H, 00=F
0029 +  0000 0041              EJ     EQU $41  EJECT COUNT
0030 +  0000 0042              PS     EQU $42  PAUSE; 00=YES
0031 +  0000 0043              ES     EQU $43  ESCAPE CHAR
0032 +  0000 0044              LDP    EQU $44  DEPTH LINES/PAGE
0033 +  0000 0045              LDPCNT EQU $45  DEPTH TEMP
0034 +  0000 0046              LWD    EQU $46  WIDTH CHARS/LINE
0035                          *
0036    0000 CE 0053 R START  LDX #PRMPT1   PROMPT FOR DRIVE
0037                          PRTMSG
0038 +  0003 3F               SWI
0039 +  0004 31               FCB 49
0040                          GTCMD         GET USER RESPONSE
0041 +  0005 3F               SWI
0042 +  0006 30               FCB 48
0043    0007 D6 25            LDA B RC      CHECK TOKEN
0044    0009 C1 03            CMP B #3      NUMBER?
0045    000B 26 22            BNE NOTNUM    NO
0046                          *
0047    000D 7D 0027          TST VALUE     NUMBER TOO BIG?
0048    0010 26 28            BNE BADNUM    YES, ERROR
0049                          *
0050    0012 96 28            LDA A VALUE+1 NUMBER TOO BIG?
0051    0014 81 03            CMP A #3      (4 DRIVES FOR SWTPC)
0052    0016 22 22            BHI BADNUM    YES, ERROR
0053                          *
0054    0018 CE 0063 R        LDX #PRMPT2   ISSUE SECOND PROMPT
0055    001B 8B 30            ADD A #$30    MAKE DRIVE NO. ASCII
0056    001D B7 0069 R        STA A DNUM
0057                          PRTMSG
0058 +  0020 3F               SWI
0059 +  0021 31               FCB 49
0060                          GTCMD         GET RESPONSE
```

```
0061 +  0022 3F                SWI
0062 +  0023 30                FCB 48
0063    0024 DE 20              LDX DESCRA
0064    0026 A6 00              LDA A 0,X
0065    0028 81 59              CMP A #'Y       WAS RESPONSE "YES"?
0066    002A 26 D4              BNE START       IF NOT, RETRY
0067                    *
0068    002C 7E 0073 R          JMP FORM2       IF SO, BEGIN FORMATTING
0069                    *
0070    002F DE 20      NOTNUM  LDX DESCRA      CHECK FOR "ESCAPE"
0071    0031 A6 00              LDA A 0,X
0072    0033 91 43              CMP A ES
0073    0035 26 03              BNE BADNUM      IF NOT, ERROR
0074                    *
0075                            INITDK          RE-INIT. DISK DRIVES
0076 +  0037 3F                SWI
0077 +  0038 33                FCB 51
0078    0039 39                 RTS             IF SO, RETURN TO SYSTEM
0079                    *
0080    003A CE 0041 R  BADNUM  LDX #BADMSG     ERROR MESSAGE
0081                            PRTMSG
0082 +  003D 3F                SWI
0083 +  003E 31                FCB 49
0084    003F 20 BF              BRA START       RETRY
0085                    *
0086    0041 20         BADMSG  FCC ' BAD DRIVE NUMBER'
0087    0052 0D                 FCB $0D
0088                    *
0089    0053 0A         PRMPT1  FCB $0A
0090    0054 44                 FCC 'DRIVE NUMBER? '
0091    0062 04                 FCB $04
0092                    *
0093    0063 44         PRMPT2  FCC 'DRIVE '
0094    0069 0001       DNUM    RMB 1
0095    006A 20                 FCC ' READY? '
0096    0072 04                 FCB $04
0098                    * FORMAT DISK HERE
0099                    *
0100    0073 7F 008A R  FORM2   CLR TRACK       START AT TRACK 0
0101                    *
0102    0076 BD 0092 R  FORM2A  JSR TRKBLD      BUILD TRACK IMAGE
0103    0079 BD 0103 R          JSR TRKWRT      WRITE TRACK IMAGE TO DISK
0104    007C B6 008A R          LDA A TRACK
0105    007F 4C                 INC A
0106    0080 B7 008A R          STA A TRACK     BUMP TRACK
0107    0083 81 23              CMP A #35       DONE?
0108    0085 26 EF              BNE FORM2A      LOOP UNTIL DONE
0109                    *
0110    0087 7E 0000 R          JMP START       BACK TO BEGINNING
0111                    *
0112    008A 0001       TRACK   RMB 1
0113    008B 0001       SECTOR  RMB 1
0114    008C 0002       SAVEX   RMB 2           DUMMY RCB (FOR "PRTERR")
0115    008E 46                 FCC 'FMT'       'FMT' ERROR
0116    0091 0001       ERRCOD  RMB 1           DUMMY "FCBSTA"
0118                    * FORM A TRACK IMAGE IN MEMORY "TRKBUF" (3400 BYTES)
0119                    *
0120    0092 CE 018C R  TRKBLD  LDX #TRKBUF     POINT TO BUFFER
0121    0095 86 FF              LDA A #$FF
0122    0097 C6 08              LDA B #8
0123    0099 8D 61              BSR PUTBYT      8-BYTE GAP
0124    009B 86 01              LDA A #1
0125    009D B7 008B R          STA A SECTOR    START OF SECTOR LOOP
0126                    *
0127                    * LOOP FOR SECTORS 1-18
0128                    *
0129    00A0 86 FF      SECLOP  LDA A #$FF
0130    00A2 C6 07              LDA B #7
0131    00A4 8D 56              BSR PUTBYT      7-BYTE GAP
0132    00A6 4F                 CLR A
0133    00A7 C6 04              LDA B #4
0134    00A9 8D 51              BSR PUTBYT      4-BYTE SYNC
0135    00AB 86 FE              LDA A #$FE
0136    00AD A7 00              STA A 0,X       ID-ADDRESS MARK
0137    00AF 08                 INX
0138    00B0 B6 008A R          LDA A TRACK
0139    00B3 A7 00              STA A 0,X       TRACK NO.
0140    00B5 08                 INX
0141    00B6 6F 00              CLR 0,X         ZERO
0142    00B8 08                 INX
0143    00B9 B6 008B R          LDA A SECTOR
0144    00BC A7 00              STA A 0,X       SECTOR NO.
0145    00BE 08                 INX
0146    00BF 6F 00              CLR 0,X         LENGTH=128
0147    00C1 08                 INX
0148    00C2 86 F7              LDA A #$F7
0149    00C4 A7 00              STA A 0,X       CRC
0150    00C6 08                 INX
0151    00C7 86 FF              LDA A #$FF
0152    00C9 C6 0B              LDA B #11
0153    00CB 8D 2F              BSR PUTBYT      11-BYTE ID-GAP
0154    00CD 4F                 CLR A
0155    00CE C6 06              LDA B #6
0156    00D0 8D 2A              BSR PUTBYT      6-BYTE SYNC
0157    00D2 86 FB              LDA A #$FB
0158    00D4 A7 00              STA A 0,X       DATA-ADDRESS MARK
0159    00D6 08                 INX
0160    00D7 4F                 CLR A
0161    00D8 C6 80              LDA B #128
0162    00DA 8D 20              BSR PUTBYT      128 BYTES OF DATA (0000)
0163    00DC 86 F7              LDA A #$F7
0164    00DE A7 00              STA A 0,X       CRC
0165    00E0 08                 INX
0166    00E1 86 FF              LDA A #$FF
0167    00E3 A7 00              STA A 0,X       PAD
0168    00E5 08                 INX
0169                    *
0170                    * END OF SECTOR DATA
0171                    *
0172    00E6 B6 008B R          LDA A SECTOR
0173    00E9 4C                 INC A
0174    00EA B7 008B R          STA A SECTOR    BUMP SECTOR
0175    00ED 81 13              CMP A #19       DONE?
0176    00EF 26 AF              BNE SECLOP      LOOP THROUGH 18 SECTORS
0177                    *
0178                    * FINISH OUT TRACK WITH LONG GAP
0179                    *
0180    00F1 86 FF              LDA A #$FF
0181    00F3 C6 C8              LDA B #200
0182    00F5 8D 05              BSR PUTBYT
0183    00F7 C6 C8              LDA B #200
0184    00F9 8D 01              BSR PUTBYT
```

```
0185   00FB 39                RTS             DONE!
0186                  *
0187                  *
0188   00FC A7 00     PUTBYT STA A 0,X       PUT BYTE INTO TRKBUF
0189   00FE 08                INX
0190   00FF 5A                DEC B           DONE?
0191   0100 26 FA             BNE PUTBYT      LOOP ON COUNT IN "B"
0192                  *
0193   0102 39                RTS
0195                  * WRITE TRACK IMAGE TO DISK DRIVE
0196                  * IMAGE IN "TRKBUF", DRIVE NO. IN "VALUE+1"
0197                  *
0198   0103 8014      DRVREG EQU $8014
0199   0103 8018      CMDREG EQU $8018
0200   0103 801B      DATREG EQU $801B
0201                  *
0202   0103 F6 8018   TRKWRT LDA B CMDREG    TURN DRIVES ON
0203   0106 C4 80             AND B #$80      READY?
0204   0108 27 20             BEQ TRKW2       YES
0205                  *
0206   010A CE 0000           LDX #0          NO, LONG DELAY
0207   010D 09        T1      DEX
0208   010E 26 FD             BNE T1
0209                  *
0210   0110 CE 0000           LDX #0          ANOTHER DELAY
0211   0113 09        T2      DEX
0212   0114 26 FD             BNE T2
0213                  *
0214   0116 F6 8018           LDA B CMDREG    READY NOW?
0215   0119 C4 80             AND B #$80
0216   011B 27 0D             BEQ TRKW2       YES
0217                  *
0218                  * DISK ERRORS HANDLED HERE
0219                  *
0220   011D F7 0091 R DSKERR STA B ERRCOD
0221   0120 CE 008C R         LDX #SAVEX
0222                          PRTERR          ISSUE ERROR MESSAGE
0223 +  0123 3F                SWI
0224 +  0124 1E                FCB 30
0225   0125 31                INS
0226   0126 31                INS             CLEAN STACK (JSR TRKWRT)
0227   0127 7E 0000 R         JMP START       RETRY
0228                  *
0229                  *
0230   012A 0F        TRKW2   SEI             DISABLE INTERRUPTS
0231   012B 96 28             LDA A VALUE+1   GET DRIVE NUMBER
0232   012D B7 8014           STA A DRVREG    SET DRIVE NUMBER
0233   0130 8D 47             BSR DEL30U      30 USEC DELAY
0234   0132 B6 008A R         LDA A TRACK
0235   0135 B7 801B           STA A DATREG    SET TRACK NO.
0236   0138 8D 3F             BSR DEL30U      30 USEC DELAY
0237   013A 86 1B             LDA A #$1B
0238   013C B7 8018           STA A CMDREG    "SEEK TRACK" COMMAND
0239   013F 8D 38             BSR DEL30U      30 USEC DELAY
0240   0141 B6 8018   TRKW3   LDA A CMDREG
0241   0144 85 01             BIT A #1        WAIT ON BUSY
0242   0146 26 F9             BNE TRKW3
0243                  *
0244   0148 BF 008C R         STS SAVEX       SAVE STACK POINTER
0245   014B 8E 018B R         LDS #TRKBUF-1   POINT TO TRACK IMAGE
0246   014E 86 F4             LDA A #$F4
0247   0150 B7 8018           STA A CMDREG    "WRITE TRACK" COMMAND
0248   0153 08                INX
0249   0154 09                DEX
0250   0155 08                INX
0251   0156 09                DEX
0252   0157 08                INX
0253   0158 09                DEX             DELAY
0254   0159 08                INX
0255   015A 09                DEX
0256                  *
0257   015B B6 8018   TRKLOP LDA A CMDREG    CHECK STATUS BITS
0258   015E 84 03             AND A #$03      DRQ AND BUSY BITS
0259   0160 88 01             EOR A #$01      INVERT BUSY BIT
0260   0162 27 F7             BEQ TRKLOP
0261                  *
0262   0164 85 02             BIT A #$02      DATA REQUEST?
0263   0166 26 02             BNE TRKRQT      YES
0264                  *
0265   0168 20 06             BRA TRKDON      OTHERWISE, FDC DONE
0266                  *
0267   016A 33        TRKRQT PUL B           GET DATA BYTE
0268   016B F7 801B           STA B DATREG    WRITE BYTE
0269   016E 20 EB             BRA TRKLOP      LOOP UNTIL DONE
0270                  *
0271   0170 BE 008C R TRKDON LDS SAVEX       RECOVER STACK POINTER
0272   0173 F6 8018           LDA B CMDREG.   CHECK FOR ERRORS
0273   0176 26 A5             BNE DSKERR
0274                  *
0275   0178 39                RTS             DONE!
0276                  *
0277                  * 30 MICROSECOND (APPROX.) DELAY FOR COMMAND
0278                  *
0279   0179 08        DEL30U INX
0280   017A 09                DEX
0281   017B 08                INX
0282   017C 09                DEX
0283   017D 08                INX
0284   017E 09                DEX
0285   017F 08                INX
0286   0180 09                DEX
0287   0181 39                RTS
0288                  *
0289                  *
0290   0182 018C    R TRKBUF EQU *+10       START OF TRACK IMAGE
0291                  *
0292                          END
```

```
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BADMSG 0041 R
BADNUM 003A R
BASEQU 2A2A M
BMEM   0033
BS     0039
CHAIN  243A M
CLASS  0026
CLOSE  2369 M
CMDREG 8018
CMEM   0037
CMPC   231B M
CMWC   2572 M
CUCHAR 0023
DATREG 801B
DEL30U 0179 R
DELETE 2420 M
DESCRA 0020
DESCRC 0022
DIV16  2524 M
DL     003A
DNUM   0069 R
DP     003B
DPCNT  003C
DRVREG 8014
DSKERR 011D R
DX     0040
EJ     0041
EMEM   0035
ERRCOD 0091 R
ES     0043
FCBCHN 0029
FCBDEF 2650 M
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
FORM2  0073 R
FORM2A 0076 R
FORMAT 0000 RN
FRETAB 002B
GETDR  23EC M
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
IOHDR  2335 M
LDP    0044
LDPCNT 0045
LOADB  246E M
LWD    0046
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NL     003E
NOTNUM 002F R
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
PRMPT1 0053 R
PRMPT2 0063 R
PRTERR 2454 M
PRTMSG 250A M
PS     0042
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTBYT 00FC R
PUTDR  2406 M
RC     0025
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
SAVEX  008C R
SECLOP 00A0 R
SECTOR 008B R
START  0000 R
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
T1     010D R
T2     0113 R
TABX   219C M
TB     003F
TRACK  008A R
TRKBLD 0092 R
TRKBUF 018C R
TRKDON 0170 R
TRKLOP 015B R
TRKRQT 016A R
TRKW2  012A R
TRKW3  0141 R
TRKWRT 0103 R
TXAB   2183 M
VALUE  0027
WD     003D
WRITE  23D2 M
XABX   21B5 M
```

```
0001  0000 0000  N        NAM SMOKEDRV
0002                     *
0003                     * DISK DRIVERS FOR SMOKE-SIGNALS BFD-68 AND CP/68 SYSTEM
0004                     * USES INFO. FROM D. L. PUCKETT'S ARTICLE IN "68" JOURNAL
0005                     * VOL. 1 ISSUE 1  PAGES 13-20
0006                     *
0007                     * BFD-68 ROM INTERFACE ADDRESSES
0008                     *
0009  0000 7E 8026   INITP  JMP $8026      INITIALIZE INTERFACE PIA
0010  0003 7E 8029   READS  JMP $8029      READ DISK SECTOR
0011  0006 7E 802C   WRITES JMP $802C      WRITE A DISK SECTOR
0012  0009 7E 8038   RESTOR JMP $8038      SEEK TRACK ZERO
0013  000C 7E 806C   WTRKR  JMP $806C      WRITE TRACK REG. OF FDC
0014  000F 7E 8072   RDTRKR JMP $8072      READ TRACK REG. OF FDC
0015                     *
0016                     * BFD-68 RAM ADDRESSES
0017                     *
0018  0012 A07B      DRIVE  EQU $A07B
0019  0012 A07C      TRACK  EQU $A07C
0020  0012 A07D      SECTOR EQU $A07D
0021  0012 A07E      BUFPNT EQU $A07E
0022                     *
0023                     * CP/68 FCB DEFINITIONS
0024                     *
0025  0012 0005      FCBSTA EQU 5          STATUS
0026  0012 0007      FCBDBA EQU 7          BUFFER ADDRESS
0027  0012 0009      FCBDRV EQU 9          DRIVE NUMBER
0028  0012 000A      FCBTRK EQU 10         TRACK
0029  0012 000B      FCBSCT EQU 11         SECTOR
0030                     *
0031                     * USER REGISTER DEFINITIONS
0032                     *
0033  0012 0006      UA     EQU 6          "A" REGISTER ON STACK
0034  0012 0007      UXH    EQU 7          "X"
0035                     *
0036                     * BASE-PAGE TRACK TABLE
0037                     *
0038  0012 0000      CTRK0  EQU 0
0039  0012 0001      CTRK1  EQU 1
0040  0012 0002      CTRK2  EQU 2
0041  0012 0003      CTRK3  EQU 3
0042  0012 0004      SAVEX  EQU 4
0043                     *
0044                     * ENTRY POINTS TO DRIVERS
0045                     *
0046  0012 0012  N        ENT @INTDK
0047  0012 001E  N        ENT .RDSEC
0048  0012 008F  N        ENT .WTSEC
0049                     *
0050                     * INITIALIZE DISKS
0051                     *
0052  0012 86 FF     @INTDK LDA A #$FF
0053  0014 97 00            STA A CTRK0          INIT. TRACK TABLE
0054  0016 97 01            STA A CTRK1
0055  0018 97 02            STA A CTRK2
0056  001A 97 03            STA A CTRK3
0057  001C 20 E2            BRA INITP            INIT. INTERFACE
0058                     *
0059                     *
0060                     * READ A DISK SECTOR
```

```
                        *
                RDSEC   TABX                    GET FCB ADDRESS
                        SWI
                        FCB 3
0020 DF 04              STX SAVEX               SAVE FCB ADDRESS
0022 A6 09              LDA A FCBDRV,X          GET DRIVE NO.
0024 84 03              AND A #$03              LIMIT RANGE
0026 8D 4A              BSR SETDRV              REFORMAT FOR BFD-68
0028 CE 0000            LDX #CTRK0              POINT TO TRACK TABLE
                        ADDAX                   ADD IN OFFSET
                        SWI
                        FCB 9
002D E6 00              LDA B 0,X               GET ENTRY
002F C1 FF              CMP B #$FF              INITIALIZED?
0031 26 10              BNE RDSEC1              YES
                        *
                        * RESTORE DRIVE TO TRACK 0
                        *
                        PSHX                    SAVE TABLE POINTER
                        SWI
                        FCB 5
0035 BD 000F R          JSR RDTRKR              READ FDC TRACK REGISTER
0038 F7 A07C            STA B TRACK             PUT CURRENT TRACK IN PLACE
003B BD 0009 R          JSR RESTOR              SEEK TRACK 0
                        PULX                    RECOVER TABLE POINTER
                        SWI
                        FCB 6
0040 5F                 CLR B
0041 E7 00              STA B 0,X               INIT. TABLE ENTRY
                        *
                RDSEC1  PSHX                    SAVE TABLE POINTER
                        SWI
                        FCB 5
0045 F7 A07C            STA B TRACK             SET TRACK OF DRIVE
0048 BD 000C R          JSR WTRKR               INTO FDC
004B DE 04              LDX SAVEX               POINT TO FCB
004D A6 0A              LDA A FCBTRK,X          GET TRACK
004F E6 0B              LDA B FCBSCT,X          GET SECTOR
0051 EE 07              LDX FCBDBA,X            GET BUFFER ADDRESS
0053 B7 A07C            STA A TRACK             SET UP IN BFD-68 PLACES
0056 F7 A07D            STA B SECTOR
0059 FF A07E            STX BUFPNT
                        PULX                    RECOVER TABLE POINTER
                        SWI
                        FCB 6
005E A7 00              STA A 0,X               PUT IN TRACK
0060 BD 0003 R          JSR READS               CALL "READ SECTOR"
0063 5D         DONE    TST B                   ERRORS?
0064 27 02              BEQ DONE2               NO
                        *
0066 C6 05              LDA B #5                YES, ERROR NO. 5
                        *
0068 30         DONE2   TSX
0069 E7 06              STA B UA,X              RETURN STATUS IN "A"
006B DE 04              LDX SAVEX               POINT TO FCB
006D EA 05              ORA B FCBSTA,X          ADD IN STATUS
006F E7 05              STA B FCBSTA,X
0071 39                 RTS
                        *
                        *
                        * REFORMAT DRIVE NO.  FOR BFD-68
                        *
                        * DRIVE 0=08 HEX
                        *       1=10
                        *       2=20
                        *
0072 4D         SETDRV  TST A                   DRIVE 0?
0073 26 04              BNE SET1                NO
                        *
0075 C6 08              LDA B #$08              YES
0077 20 12              BRA SETD
                        *
0079 81 01      SET1    CMP A #1                DRIVE 1?
007B 26 04              BNE SET2                NO
                        *
007D C6 10              LDA B #$10              YES
007F 20 0A              BRA SETD
                        *
0081 81 02      SET2    CMP A #2                DRIVE 2?
0083 26 04              BNE SET3                NO
                        *
0085 C6 20              LDA B #$20              YES
0087 20 02              BRA SETD
                        *
0089 C6 08      SET3    LDA B #$08              DEFAULT TO DRIVE 0
                        *
008B F7 A07B    SETD    STA B DRIVE             SET DRIVE IN PLACE
008E 39                 RTS
                        *
                        *
                        * WRITE A DISK SECTOR
                        *
                WTSEC   TABX                    POINT TO FCB
                        SWI
                        FCB 3
0091 DF 04              STX SAVEX
0093 A6 09              LDA A FCBDRV,X          GET DRIVE NO.
0095 84 03              AND A #$03              LIMIT RANGE
0097 8D D9              BSR SETDRV              RE-FORMAT DRIVE NO.
0099 CE 0000            LDX #CTRK0              POINT TO TRACK TABLE
                        ADDAX                   ADD IN OFFSET
                        SWI
                        FCB 9
009E E6 00              LDA B 0,X               GET ENTRY
00A0 C1 FF              CMP B #$FF              INITIALIZED?
00A2 26 10              BNE WTSEC1              YES
                        *
                        * RESTORE DRIVE TO TRACK 0
                        *
                        PSHX                    SAVE TABLE POINTER
                        SWI
                        FCB 5
00A6 BD 000F R          JSR RDTRKR              READ FDC TRACK REGISTER
00A9 F7 A07C            STA B TRACK             SAVE CURRENT TRACK
00AC BD 0009 R          JSR RESTOR              SEEK TRACK 0
                        PULX                    RECOVER TABLE POINTER
                        SWI
                        FCB 6
00B1 5F                 CLR B
00B2 E7 00              STA B 0,X               INIT. TABLE ENTRY
                        *
                WTSEC1  PSHX                    SAVE TABLE ENTRY
```

```
0183 +  00B4 3F              SWI
0184 +  00B5 05              FCB 5
0185    00B6 F7 A07C         STA B TRACK          SET TRACK OF DRIVE
0186    00B9 BD 000C R       JSR WTRKR            INTO FDC
0187    00BC DE 04           LDX SAVEX            POINT TO FCB
0188    00BE A6 0A           LDA A FCBTRK,X       GET TRACK
0189    00C0 E6 0B           LDA B FCBSCT,X       GET SECTOR
0190    00C2 EE 07           LDX FCBDBA,X         GET BUFFER ADDRESS
0191    00C4 B7 A07C         STA A TRACK          PUT INTO BFD-68 PLACES
0192    00C7 F7 A07D         STA B SECTOR
0193    00CA FF A07E         STX BUFPNT
0194                         PULX                 RECOVER TABLE POINTER
0195 +  00CD 3F              SWI
0196 +  00CE 06              FCB 6
0197    00CF A7 00           STA A 0,X            NEW ENTRY
0198    00D1 BD 0006 R       JSR WRITES           CALL "WRITE SECTOR"
0199    00D4 20 8D           BRA DONE             ERROR CHECK AND FINISH
0200                 *
0201                         END
```

```
.RDSEC  001E RN
.WTSEC  008F RN
@INTDK  0012 RN
ADDABX  2219 M
ADDAX   2232 M
ADDBX   224B M
ADDXAB  2200 M
BASEQU  2A2A M
BUFPNT  A07E
CHAIN   243A M
CLOSE   2369 M
CMPC    231B M
CMWC    2572 M
CTRK0   0000
CTRK1   0001
CTRK2   0002
CTRK3   0003
DELETE  2420 M
DIV16   2524 M
DONE    0063 R
DONE2   0068 R
DRIVE   A07B
FCBDBA  0007
FCBDEF  2650 M
FCBDRV  0009
FCBSCT  000B
FCBSTA  0005
FCBTRK  000A
FIBDEF  2940 M
FMTFCB  2488 M
FMTS    2558 M
GETDR   23EC M
GTCMD   24F0 M
INDEX   24BC M
INITDK  253E M
INITP   0000 R
IOHDR   2335 M
LOADB   246E M
MOVC    2301 M
MOVS    24A2 M
MUL16   22E7 M
MUL8    22CD M
NXTOK   24D6 M
OPEN    234F M
OPEND   239E M
PRTERR  2454 M
PRTMSG  250A M
PSHALL  2151 M
PSHX    21CE M
PULLAL  216A M
PULX    21E7 M
PUTDR   2406 M
RCBDEF  258C M
RDSEC1  0043 R
RDTRKR  000F R
READ    23B8 M
READS   0003 R
RESTOR  0009 R
REWIND  2384 M
SAVEX   0004
SECTOR  A07D
SET1    0079 R
SET2    0081 R
SET3    0089 R
SETD    008B R
SETDRV  0072 R
SMOKED  0000 RN
SUBABX  227F M
SUBAX   2299 M
SUBBX   22B3 M
SUBXAB  2265 M
TABX    219C M
TRACK   A07C
TXAB    2183 M
UA      0006
UXH     0007
WRITE   23D2 M
WRITES  0006 R
WTRKR   000C R
WTSEC1  00B4 R
XABX    21B5 M
```

```
0001   0000 0000    N        NAM BOOT
0002                 *
0003                 * SMOKE-SIGNALS CP/68 BOOTSTRAP PROGRAM
0004                 * ASSUMES SYSTEM FILE LINKED AS FOLLOWS:
0005                 *
0006                 *      TRACK 0, SECTOR 1, BYTE 122-FIRST TRACK
0007                 *                             123-FIRST SECTOR
0008                 *                             124-LAST TRACK
0009                 *                             125-LAST SECTOR
0010                 *                             126,7 FREE-SPACE HEADER
0011                 *
0012                 * BOOTS SYSTEM FROM DRIVE 0:
0013                 *
0014                 * DEFINE DISK-DRIVE INTERFACE ADDRESSING
0015                 *
0016   0000 A07B     DRIVE  EQU $A07B
0017   0000 A07C     TRACK  EQU $A07C
0018   0000 A07D     SECTOR EQU $A07D
0019   0000 A07E     BUFPNT EQU $A07E
0020                 *
0021   0000 8026     INITP  EQU $8026      INIT. INTERFACE PIA
0022   0000 8029     READS  EQU $8029      READ DISK SECTOR
0023   0000 8038     RESTOR EQU $8038      SEEK TRACK 0
0024   0000 8072     RDTRKR EQU $8072      READ FDC TRACK REG.
0025                 *
0026                 * NOTE: ALL VARIABLES IN COMMON, CODE IS ROM-ABLE
0027                 *
0028   0000 0000   C        CMN STACK,16
0029   0000 0010   C        CMN BUFFER,128
0030   0000 0090   C        CMN FTS,2
0031   0000 0092   C        CMN LTS,2
0032   0000 0094   C        CMN PTS,2
0033   0000 0096   C        CMN INDEX,2
0034   0000 0098   C        CMN SAVEX,2
0035   0000 009A   C        CMN SAVEX2,2
0036   0000 009C   C        CMN ADDRES,2
0037   0000 009E   C        CMN FCNT,1
0038   0000 009F   C        CMN RCNT,1
0039                 *
0040                 * ERROR JUMP VECTOR
0041                 *
0042   0000 7E E113  ERROR  JMP $E113
0043                 *
0044                 * BEGIN BOOT HERE
0045                 *
0046   0003 8E 000F C START  LDS #STACK+15 INIT. STACK POINTER
0047   0006 86 08           LDA A #$08    DRIVE 0 IN BFD-FORMAT
0048   0008 B7 A07B         STA A DRIVE
0049   000B BD 8026         JSR INITP     INIT. INTERFACE
0050   000E BD 8072         JSR RDTRKR    GET TRACK FROM FDC
0051   0011 F7 A07C         STA B TRACK
0052   0014 BD 8038         JSR RESTOR    SEEK TRACK 0
0053                 *
0054                 * NOW GET SYSTEM LINK INFORMATION
0055                 *
0056   0017 86 01           LDA A #1
0057   0019 C6 00           LDA B #0      TRACK 0, SECTOR 1
0058   001B CE 0010 C       LDX #BUFFER
0059   001E BD 00C6 R       JSR RDSEC     READ LINK SECTOR
0060   0021 CE 0010 C       LDX #BUFFER
0061   0024 A6 7A           LDA A 122,X   GET FIRST T/S
0062   0026 E6 7B           LDA B 123,X
0063   0028 B7 0090 C       STA A FTS
0064   002B F7 0091 C       STA B FTS+1
0065   002E A6 7C           LDA A 124,X   GET LAST T/S
0066   0030 E6 7D           LDA B 125,X
0067   0032 B7 0092 C       STA A LTS
0068   0035 F7 0093 C       STA B LTS+1
0069   0038 CE 0014 C       LDX #BUFFER+4
0070   003B FF 0096 C       STX INDEX     INIT. BUFFER INDEX
0071   003E B6 0091 C       LDA A FTS+1
0072   0041 F6 0090 C       LDA B FTS
0073   0044 B7 0095 C       STA A PTS+1   INIT. PRESENT T/S
0074   0047 F7 0094 C       STA B PTS
0075   004A CE 0010 C       LDX #BUFFER
0076   004D BD 00C6 R       JSR RDSEC     READ FIRST SECTOR
0077                 *
0078                 * NOW LOAD SYSTEM FILE INTO MEMORY
0079                 *
0080   0050 8D 3A    BOOT1  BSR GETBYT    GET A DATA BYTE FROM FILE
0081   0052 81 16           CMP A #$16    TRANSFER-ADDRESS?
0082   0054 26 0C           BNE BOOT2     NO
0083                 *
0084   0056 8D 34           BSR GETBYT
0085   0058 B7 009C C       STA A ADDRES  GET TRANSFER ADDRESS
0086   005B 8D 2F           BSR GETBYT
0087   005D B7 009D C       STA A ADDRES+1
0088   0060 20 EE           BRA BOOT1     GET NEW DATA FRAME
0089                 *
0090   0062 81 02    BOOT2  CMP A #$02    DATA FRAME?
0091   0064 26 21           BNE BOOT4     NO
0092                 *
0093   0066 8D 24           BSR GETBYT
0094   0068 B7 0098 C       STA A SAVEX   GET ADDRESS
0095   006B 8D 1F           BSR GETBYT
0096   006D B7 0099 C       STA A SAVEX+1
0097   0070 8D 1A           BSR GETBYT
0098   0072 B7 009E C       STA A FCNT    GET FRAME COUNTER
0099                 *
0100   0075 8D 15    BOOT3  BSR GETBYT    GET DATA BYTE
0101   0077 FE 0098 C       LDX SAVEX
0102   007A A7 00           STA A 0,X     STORE BYTE
0103   007C 08              INX
0104   007D FF 0098 C       STX SAVEX
0105   0080 7A 009E C       DEC FCNT      COUNT DOWN
0106   0083 26 F0           BNE BOOT3
0107                 *
0108   0085 20 C9           BRA BOOT1     GET NEW DATA FRAME
0109                 *
0110   0087 FE 009C C BOOT4  LDX ADDRES    GET TRANSFER ADDRESS
0111   008A 6E 00           JMP 0,X       GO THERE
0112                 *
0114                 * READ A DATA BYTE FROM SYSTEM FILE
0115                 * RETURN BYTE IN 'A' REGISTER
0116                 *
0117   008C FE 0096 C GETBYT LDX INDEX
0118   008F 8C 0090 C       CPX #BUFFER+128      NEED NEW SECTOR?
0119   0092 27 07           BEQ GETSEC           YES
0120                 *
0121   0094 A6 00           LDA A 0,X            GET BYTE
0122   0096 08              INX
```

```
0123  0097 FF 0096 C          STX INDEX             MOVE POINTER
0124  009A 39                 RTS
0125                    *
0126  009B F6 0094 C GETSEC   LDA B PTS
0127  009E B6 0095 C          LDA A PTS+1           CHECK FOR LAST SECTOR
0128  00A1 B1 0093 C          CMP A LTS+1
0129  00A4 26 07              BNE GETS2             NOT LAST
0130                    *
0131  00A6 F1 0092 C          CMP B LTS
0132  00A9 26 02              BNE GETS2             NOT LAST
0133                    *
0134  00AB 20 DA              BRA BOOT4             EOF-GO TO TRANSFER ADDRESS
0135                    *
0136  00AD CE 0010 C GETS2    LDX #BUFFER
0137  00B0 E6 00              LDA B 0,X             GET FORWARD T/S LINK
0138  00B2 A6 01              LDA A 1,X
0139  00B4 F7 0094 C          STA B PTS             UPDATE PRESENT T/S
0140  00B7 B7 0095 C          STA A PTS+1
0141  00BA 8D 0A              BSR RDSEC             READ NEW SECTOR
0142  00BC CE 0014 C          LDX #BUFFER+4
0143  00BF A6 00              LDA A 0,X             GET DATA BYTE
0144  00C1 08                 INX
0145  00C2 FF 0096 C          STX INDEX             RE-INIT. INDEX
0146  00C5 39                 RTS
0147                    *
0149                    * SINGLE-SECTOR READ ROUTINE
0150                    *
0151                    *       DRIVE=0
0152                    *       TRACK='B'
0153                    *       SECTOR='A'
0154                    *       BUFFER='X'
0155                    *
0156  00C6 F7 A07C    RDSEC   STA B TRACK
0157  00C9 B7 A07D            STA A SECTOR          PUT DATA INTO PLACE
0158  00CC FF A07E            STX BUFPNT
0159  00CF BD 8029            JSR READS             READ DATA SECTOR
0160  00D2 5D                 TST B                 ERROR?
0161  00D3 27 03              BEQ DONE              NO
0162                    *
0163  00D5 7E 0000 R          JMP ERROR             YES
0164                    *
0165  00D8 39         DONE    RTS
0166                    END
```

```
ADDRES 009C C
BOOT   0000 RN
BOOT1  0050 R
BOOT2  0062 R
BOOT3  0075 R
BOOT4  0087 R
BUFFER 0010 C
BUFPNT A07E
DONE   00D8 R
DRIVE  A07B
ERROR  0000 R
FCNT   009E C
FTS    0090 C
GETBYT 008C R
GETS2  00AD R
GETSEC 009B R
INDEX  0096 C
INITP  8026
LTS    0092 C
PTS    0094 C
RCNT   009F C
RDSEC  00C6 R
RDTRKR 8072
READS  8029
RESTOR 8038
SAVEX  0098 C
SAVEX2 009A C
SECTOR A07D
STACK  0000 C
START  0003 R
TRACK  A07C
```

```
0001    0000 0000    N       NAM INITER
0002                 *
0003                 * INITIALIZE A DISK FOR CP-68 OPERATING SYSTEM
0004                 *
0005                 * FOR SWTPC 5 INCH FLOPPY DISKS
0006                 *
0007                 * TRACK 0, SECTOR 1          BOOTSTRAP
0008                 * TRACK 0, SECTOR 1          HEADER OF FREE-SPACE LIST
0009                 * TRACK 0, SECTORS 2-18      DIRECTORY SPACE
0010                 * TRACKS 1-35                FREE-SPACE
0011                 *
0012                 * DISK ATTRIBUTES
0013                 *
0014    0000 0080    SECSIZ EQU 128        128 BYTES PER SECTOR
0015    0000 0012    TRKSIZ EQU 18         18 SECTORS PER TRACK
0016    0000 0022    DSKSIZ EQU 34         34 TRACKS ON DISK (LESS TRACK 0)
0017                 *
0018                 * FILE-CONTROL BLOCK ADDRESSES
0019                 *
0020                        FCBDEF
0021 +  0000 0000     FCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0022 +  0000 0002     FCBGDT EQU 2          GENERIC DEVICE TYPE
0023 +  0000 0005     FCBSTA EQU 5          STATUS
0024 +  0000 0006     FCBDTT EQU 6          DATA TRANSFER TYPE
0025 +  0000 0007     FCBDBA EQU 7          DATA BUFFER ADDRESS
0026 +  0000 0009     FCBDRV EQU 9          DRIVE NUMBER
0027 +  0000 000A     FCBTRK EQU 10         TRACK NUMBER
0028 +  0000 000B     FCBSCT EQU 11         SECTOR NUMBER
0029 +  0000 000C     FCBFWD EQU 12         FWD LINK TRACK/SECTOR
0030 +  0000 000E     FCBBAK EQU 14         BACK LINK TRACK/SECTOR
0031 +  0000 0010     FCBNAM EQU 16         FILE NAME (8.3+EOT=13))
0032 +  0000 001D     FCBTYP EQU 29         FILE TYPE
0033 +  0000 001E     FCBACS EQU 30         FILE ACCESS CODE
0034 +  0000 001F     FCBFTS EQU 31         FIRST TRACK/SECTOR
0035 +  0000 0021     FCBLTS EQU 33         LAST TRACK/SECTOR
0036 +  0000 0023     FCBNMS EQU 35         NUMBER OF SECTORS
0037 +  0000 0025     FCBNFB EQU 37         NEXT FCB IN ACTIVE CHAIN
0038 +  0000 0027     FCBIND EQU 39         INDEX INTO DATA BUFFER
0039 +  0000 0029     FCBSCF EQU 41         SPACE COMPRESSION FLAG
0040                 *
0041    0000 0002    FCBSPC RMB 2          FILE-CONTROL BLOCK
0042    0002 44             FCC 'DSK'      DISK
0043    0005 0001           RMB 1
0044    0006 FF             FCB $FF        OUTPUT
0045    0007 0023           RMB 35
0046                 *
0047    002A 0080    BUFFER RMB SECSIZ     SECTOR BUFFER
0048                 *
0049                 * COMMAND-LINE INTERPRETER BASE-PAGE LOCATIONS
0050                 *
0051                        BASEQU
0052 +  00AA 0020     DESCRA EQU $20  DESCRIPTOR ADDRESS(2)
0053 +  00AA 0022     DESCRC EQU $22  DESCRIPTOR COUNT
0054 +  00AA 0023     CUCHAR EQU $23  CURRENT CHAR (2)
0055 +  00AA 0025     RC     EQU $25  TOKEN RETURN CODE
0056 +  00AA 0026     CLASS  EQU $26  TOKEN CLASS
0057 +  00AA 0027     VALUE  EQU $27  BIN VALUE/TRANSFER ADDRESS (2)
0058 +  00AA 0029     FCBCHN EQU $29  TOP OF FCB CHAIN (2)
0059 +  00AA 002B     FRETAB EQU $2B  DISK FREE SPACE POINTER (8)
0060 +  00AA 0033     BMEM   EQU $33  START OF TRANSIENT AREA(2)
0061 +  00AA 0035     EMEM   EQU $35  END OF TRANSIENT AREA (2)
0062 +  00AA 0037     CMEM   EQU $37  NEXT AVAIL TRANSIENT AREA (2)
0063 +  00AA 0039     BS     EQU $39  BACKSPACE CHAR
0064 +  00AA 003A     DL     EQU $3A  DELETE LINE CHAR
0065 +  00AA 003B     DP     EQU $3B  DEPTH; LINES/PAGE
0066 +  00AA 003C     DPCNT  EQU $3C  DEPTH TEMP
0067 +  00AA 003D     WD     EQU $3D  WIDTH; CHARS/LINE
0068 +  00AA 003E     NL     EQU $3E  NULL COUNT
0069 +  00AA 003F     TB     EQU $3F  TAB CHAR
0070 +  00AA 0040     DX     EQU $40  DUPLEX; FF=H, 00=F
0071 +  00AA 0041     EJ     EQU $41  EJECT COUNT
0072 +  00AA 0042     PS     EQU $42  PAUSE; 00=YES
0073 +  00AA 0043     ES     EQU $43  ESCAPE CHAR
0074 +  00AA 0044     LDP    EQU $44  DEPTH LINES/PAGE
0075 +  00AA 0045     LDPCNT EQU $45  DEPTH TEMP
0076 +  00AA 0046     LWD    EQU $46  WIDTH CHARS/LINE
0077                 *
0078    00AA 49      PROMPT FCC 'INIT.  DISK IN DRIVE '
0079    00BE 0001    DRVNO  RMB 1
0080    00BF 20             FCC ' ? '
0081    00C2 04             FCB $04
0082                 *
0083    00C3 00C3    N      ENT .INITR     ENTRY POINT FROM CLI
0084                 *
0085    00C3 96 28   .INITR LDA A VALUE+1 GET DRIVE NUMBER
0086    00C5 84 03          AND A #$03    LIMIT RANGE (SWTPC PERMITS 4 DRIVES)
0087    00C7 CE 0000 R      LDX #FCBSPC   POINT TO FCB
0088    00CA A7 09          STA A FCBDRV,X
0089    00CC 8B 30          ADD A #$30    MAKE DRIVE NUMBER ASCII
0090    00CE B7 00BE R      STA A DRVNO   PUT IN PROMPT LINE
0091    00D1 CE 00AA R      LDX #PROMPT
0092                        PRTMSG        OUTPUT PROMPT
0093 +  00D4 3F              SWI
0094 +  00D5 31              FCB 49
0095                        GTCMD         GET USER RESPONSE
0096 +  00D6 3F              SWI
0097 +  00D7 30              FCB 48
0098    00D8 DE 20          LDX DESCRA
0099    00DA A6 00          LDA A 0,X     GET FIRST CHAR. OF RESPONSE
0100    00DC 81 59          CMP A #'Y     WAS IT 'YES'?
0101    00DE 27 01          BEQ INITR2    IF SO, CONTINUE
0102                 *
0103    00E0 39             RTS           IF NOT, QUIT
0104                 *
0105    00E1 CE 0000 R INITR2 LDX #FCBSPC   POINT TO FCB
0106    00E4 6F 0A          CLR FCBTRK,X  TRACK=0
0107    00E6 86 01          LDA A #1
0108    00E8 A7 0B          STA A FCBSCT,X  SECTOR=1
0109                 *
0110                 * INITIALIZE HEAD OF FREE-SPACE BLOCK
0111                 *
0112                 * ALL ZERO EXCEPT FOR LAST TWO BYTES=TRACK 1, SECTOR 1
0113                 *
0114                        TXAB
0115 +  00EA 3F              SWI
0116 +  00EB 02              FCB 2
0117    00EC CE 002A R      LDX #BUFFER
0118                        XABX
0119 +  00EF 3F              SWI
0120 +  00F0 04              FCB 4
0121    00F1 A7 07          STA A FCBDBA,X
```

```
00F3 E7 08            STA B FCBDBA+1,X
                      PSHX
00F5 3F                SWI
00F6 05                FCB 5
                *
                * CLEAR OUT BUFFER EXCEPT FOR LAST 2 BYTES
                *
00F7 CE 002A R        LDX #BUFFER
00FA C6 7E            LDA B #SECSIZ-2
00FC 4F               CLR A
00FD A7 00      INITR3 STA A 0,X
00FF 08               INX
0100 5A               DEC B
0101 26 FA            BNE INITR3
                *
0103 86 01            LDA A #1        TRACK, SECTOR=1
0105 A7 00            STA A 0,X
0107 A7 01            STA A 1,X
                      PULX
0109 3F                SWI
010A 06                FCB 6
010B 8D 7E            BSR WRTBLK      WRITE BLOCK 3
010D 6D 05            TST FCBSTA,X    CHECK FOR DISK ERROR
010F 27 04            BEQ *+6         OK
                *
0111 20 4F            BRA INITQ       FATAL DISK ERROR, QUIT
                *
0113 20 76      @WRTBL BRA WRTBLK     OUT OF RANGE "BSR WRTBLK"
0115 6C 0B            INC FCBSCT,X    SECTOR=4
0117 7F 00A8 R        CLR BUFFER+SECSIZ-2
011A 7F 00A9 R        CLR BUFFER+SECSIZ-1
                *
                * INITIALIZE DIRECTORY TO ZERO
                *
011D 8D 6C      INITR4 BSR WRTBLK     WRITE DIRECTORY BLOCK
011F 6D 05            TST FCBSTA,X    CHECK FOR DISK ERROR
0121 27 02            BEQ *+4         OK
                *
0123 20 3D            BRA INITQ       FATAL DISK ERROR, QUIT
                *
0125 A6 0B            LDA A FCBSCT,X
0127 4C               INC A           NEXT SECTOR
0128 81 12            CMP A #TRKSIZ DONE WITH TRACK?
012A 27 04            BEQ INITR5      YES
                *
012C A7 0B            STA A FCBSCT,X
012E 20 ED            BRA INITR4      NO, CONTINUE WRITING
                *
0130 86 01      INITR5 LDA A #1
0132 A7 0B            STA A FCBSCT,X  SECTOR=1
0134 A7 0A            STA A FCBTRK,X  TRACK=1
0136 16               TAB
                *
                * INITIALIZE REST OF DISK (FREE-SPACE)
                *
                *       X=FCB ADDRESS
                *       A=TRACK NUMBER
                *       B=SECTOR NUMBER
                *
0137 5C         INITR6 INC B          MAKE SECTOR LINKAGE
0138 C1 13            CMP B #TRKSIZ+1 END OF TRACK?
013A 26 09            BNE INITR7      NO
                *
013C C6 01            LDA B #1        YES, SECTOR=1
013E 4C               INC A           NEXT TRACK
013F 81 23            CMP A #DSKSIZ+1 END OF DISK?
0141 26 02            BNE INITR7      NO
                *
0143 4F               CLR A           LAST SECTOR POINTS TO 0,0
0144 5F               CLR B
                *
0145 B7 002A R INITR7 STA A BUFFER    TRACK LINK
0148 37               PSH B           SAVE LSEC
0149 8D 33            BSR GETSC       GET PSEC
014B F7 002B R        STA B BUFFER+1 SECTOR LINK
014E 33               PUL B           RESTORE LSEC
014F 8D 3A            BSR WRTBLK      WRITE SECTOR
0151 4D               TST A           DONE? (=0)
0152 26 04            BNE INITR8      NO
                *
0154 5D               TST B           DONE? (=0)
0155 26 01            BNE INITR8      NO
                *
0157 39               RTS             YES, DONE!!!
                *
0158 A7 0A      INITR8 STA A FCBTRK,X
015A 37               PSH B           SAVE LSEC
015B 8D 21            BSR GETSC       GET PSEC
015D E7 0B            STA B FCBSCT,X
015F 33               PUL B           GET LSEC
0160 20 D5            BRA INITR6      KEEP WRITING
                *
                *
                * FATAL ERROR MESSAGE
                *
0162 CE 0168 R INITQ  LDX #QMSG       OUTPUT ERROR MESSAGE
                      PRTMSG
0165 3F                SWI
0166 31                FCB 49
0167 39               RTS             RETURN TO CLI
                *
0168 49         QMSG  FCC 'INITIALIZATION FAILED'
017D 0D               FCB $0D
                *
                * CONVERT LSEC TO PSEC
                * LSEC IN B-REG
                *
                GETSC  PSHX           SAVE X-REGISTER
017E 3F                SWI
017F 05                FCB 5
0180 CE 01FF R        LDX #TBL        POINT TO LOGICAL/PHYSICAL TABLE
                      ADDBX           ADD LOGICAL OFFSET
0183 3F                SWI
0184 0A                FCB 10
0185 09               DEX             SECTOR STARTS AT 1
0186 E6 00            LDA B 0,X       GET PSEC
                      PULX            RESTORE X-REG
0188 3F                SWI
0189 06                FCB 6
018A 39               RTS
                *
                * WRITE A SECTOR WITH ERROR CHECKING
```

```
0244                    *
0245   018B 36          WRTBLK PSH A          SAVE 'A'
0246   018C 4F                 CLR A
0247   018D 6F 05               CLR FCBSTA,X  CLEAR ERROR FLAG
0248                            IOHDR         ISSUE I/O REQUEST
0249 +  018F 3F                  SWI
0250 +  0190 13                  FCB 19
0251   0191 A7 05               STA A FCBSTA,X
0252   0193 4D                  TST A         ERROR?
0253   0194 26 02               BNE WRTERR    YES
0254                    *
0255   0196 32                  PUL A         RESTORE 'A'
0256   0197 39                  RTS
0257                    *
0258   0198 16          WRTERR TAB
0259   0199 8D 54               BSR OUTHL     CONVERT LEFT DIGIT
0260   019B B7 01D5 R           STA A ERTYPE
0261   019E 17                  TBA
0262   019F 8D 52               BSR OUTHR     CONVERT RIGHT DIGIT
0263   01A1 B7 01D6 R           STA A ERTYPE+1
0264                            PSHX          SAVE X
0265 +  01A4 3F                  SWI
0266 +  01A5 05                  FCB 5
0267   01A6 A6 0B               LDA A FCBSCT,X
0268   01A8 8D 45               BSR OUTHL     MAKE SECTOR NO. HEX
0269   01AA B7 01E2 R           STA A SECT
0270   01AD A6 0B               LDA A FCBSCT,X
0271   01AF 8D 42               BSR OUTHR
0272   01B1 B7 01E3 R           STA A SECT+1
0273   01B4 A6 0A               LDA A FCBTRK,X
0274   01B6 8D 37               BSR OUTHL     MAKE TRACK NO. HEX
0275   01B8 B7 01EC R           STA A TRACK
0276   01BB A6 0A               LDA A FCBTRK,X
0277   01BD 8D 34               BSR OUTHR
0278   01BF B7 01ED R           STA A TRACK+1
0279   01C2 CE 01CA R           LDX #DERROR
0280                            PRTMSG        PRINT ERROR MESSAGE
0281 +  01C5 3F                  SWI
0282 +  01C6 31                  FCB 49
0283   01C7 3F                  SWI           CALL CP/68
0284   01C8 1F                  FCB 31        "WARMSTART"
0285   01C9 39                  RTS
0286                    *
0287   01CA 44          DERROR FCC 'DISK ERROR:'
0288   01D5 0002        ERTYPE RMB 2
0289   01D7 20                 FCC ' AT SECTOR '
0290   01E2 0002        SECT   RMB 2
0291   01E4 2C                 FCC ', TRACK '
0292   01EC 0002        TRACK  RMB 2
0293   01EE 0D                 FCB $0D
0294                    *
0295                    * CONVERT BINARY TO HEX-ASCII HERE
0296                    *
0297   01EF 44          OUTHL  LSR A          SHIFT RIGHT
0298   01F0 44                 LSR A
0299   01F1 44                 LSR A
0300   01F2 44                 LSR A
0301                    *
0302   01F3 84 0F       OUTHR  AND A #$0F     GET NIBBLE
0303   01F5 8B 30              ADD A #$30    MAKE ASCII
0304   01F7 81 39              CMP A #$39    >9?
0305   01F9 23 02              BLS *+4       NO
0306                    *
0307   01FB 8B 07              ADD A #$7     YES
0308                    *
0309   01FD 39                 RTS
0310                    *
0311                    *
0312                    * LOGICAL/PHYSICAL SECTOR TABLE
0313                    *
0314   01FE 00                 FCB 00
0315                    *
0316   01FF 01          TBL    FCB $1
0317   0200 06                 FCB $6
0318   0201 0B                 FCB $B
0319   0202 10                 FCB $10
0320   0203 03                 FCB $3
0321   0204 08                 FCB $8
0322   0205 0D                 FCB $D
0323   0206 12                 FCB $12
0324   0207 05                 FCB $5
0325   0208 0A                 FCB $A
0326   0209 0F                 FCB $F
0327   020A 02                 FCB $2
0328   020B 07                 FCB $7
0329   020C 0C                 FCB $C
0330   020D 11                 FCB $11
0331   020E 04                 FCB $4
0332   020F 09                 FCB $9
0333   0210 0E                 FCB $E
0334                    *
0335                    *
0336                    *
0337   0211 0211      R BOOT   EQU *         BOOT PROGRAM STARTS HERE
0338                    *
0339                    *
0340                            END
```

```
. INITR 00C3 RN
@WRTBL 0113 R
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BASEQU 2A2A M
BMEM   0033
BOOT   0211 R
BS     0039
BUFFER 002A R
CHAIN  243A M
CLASS  0026
CLOSE  2369 M
CMEM   0037
CMPC   231B M
CMWC   2572 M
CUCHAR 0023
DELETE 2420 M
DERROR 01CA R
DESCRA 0020
DESCRC 0022
DIV16  2524 M
DL     003A
DP     003B
DPCNT  003C
DRVNO  00BE R
DSKSIZ 0022
DX     0040
EJ     0041
EMEM   0035
ERTYPE 01D5 R
ES     0043
FCBACS 001E
FCBBAK 000E
FCBCHN 0029
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBDTT 0006
FCBEQT 0000
FCBFTS 001F
FCBFWD 000C
FCBGDT 0002
FCBIND 0027
FCBLTS 0021
FCBNAM 0010
FCBNFB 0025
FCBNMS 0023
FCBSCF 0029
FCBSCT 000B
FCBSPC 0000 R
FCBSTA 0005
FCBTRK 000A
FCBTYP 001D
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
FRETAB 002B
GETDR  23EC M
GETSC  017E R
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
INITER 0000 RN
INITQ  0162 R
INITR2 00E1 R
INITR3 00FD R
INITR4 011D R
INITR5 0130 R
INITR6 0137 R
INITR7 0145 R
INITR8 0158 R
IOHDR  2335 M
LDP    0044
LDPCNT 0045
LOADB  246E M
LWD    0046
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NL     003E
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
OUTHL  01EF R
OUTHR  01F3 R
PROMPT 00AA R
PRTERR 2454 M
PRTMSG 250A M
PS     0042
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
QMSG   0168 R
RC     0025
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
SECSIZ 0080
SECT   01E2 R
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
TABX   219C M
TB     003F
TBL    01FF R
TRACK  01EC R
TRKSIZ 0012
TXAB   2183 M
VALUE  0027
WD     003D
WRITE  23D2 M
WRTBLK 018B R
WRTERR 0198 R
XABX   21B5 M
```

```
0001    0000 0000     N        NAM FORMATTER
0002                           *
0003                           * PROGRAM TO FORMAT SOFT-SECTORED MINIFLOPPY DISKS
0004                           * ASSUMES BFD-68 HARDWARE WITH ROM AND PIA
0005                           * FOR CP/68 SYSTEM---35 TRACKS, 18 SECTORS/TRACK
0006                           *
0007                           * COPYRIGHT: 1979...HEMENWAY ASSOCIATES, BOSTON MASS.
0008                           *
0009                                  BASEQU        ESTABLISH CP/68 BASEPAGE
0010 +  0000 0020              DESCRA EQU $20  DESCRIPTOR ADDRESS(2)
0011 +  0000 0022              DESCRC EQU $22  DESCRIPTOR COUNT
0012 +  0000 0023              CUCHAR EQU $23  CURRENT CHAR (2)
0013 +  0000 0025              RC     EQU $25  TOKEN RETURN CODE
0014 +  0000 0026              CLASS  EQU $26  TOKEN CLASS
0015 +  0000 0027              VALUE  EQU $27  BIN VALUE/TRANSFER ADDRESS (2)
0016 +  0000 0029              FCBCHN EQU $29  TOP OF FCB CHAIN (2)
0017 +  0000 002B              FRETAB EQU $2B  DISK FREE SPACE POINTER (8)
0018 +  0000 0033              BMEM   EQU $33  START OF TRANSIENT AREA(2)
0019 +  0000 0035              EMEM   EQU $35  END OF TRANSIENT AREA (2)
0020 +  0000 0037              CMEM   EQU $37  NEXT AVAIL TRANSIENT AREA (2)
0021 +  0000 0039              BS     EQU $39  BACKSPACE CHAR
0022 +  0000 003A              DL     EQU $3A  DELETE LINE CHAR
0023 +  0000 003B              DP     EQU $3B  DEPTH; LINES/PAGE
0024 +  0000 003C              DPCNT  EQU $3C  DEPTH TEMP
0025 +  0000 003D              WD     EQU $3D  WIDTH; CHARS/LINE
0026 +  0000 003E              NL     EQU $3E  NULL COUNT
0027 +  0000 003F              TB     EQU $3F  TAB CHAR
0028 +  0000 0040              DX     EQU $40  DUPLEX; FF=H, 00=F
0029 +  0000 0041              EJ     EQU $41  EJECT COUNT
0030 +  0000 0042              PS     EQU $42  PAUSE; 00=YES
0031 +  0000 0043              ES     EQU $43  ESCAPE CHAR
0032 +  0000 0044              LDP    EQU $44  DEPTH LINES/PAGE
0033 +  0000 0045              LDPCNT EQU $45  DEPTH TEMP
0034 +  0000 0046              LWD    EQU $46  WIDTH CHARS/LINE
0035                           *
0036    0000 CE 0053 R START   LDX #PRMPT1   PROMPT FOR DRIVE
0037                           PRTMSG
0038 +  0003 3F                SWI
0039 +  0004 31                FCB 49
0040                           GTCMD          GET USER RESPONSE
0041 +  0005 3F                SWI
0042 +  0006 30                FCB 48
0043    0007 D6 25             LDA B RC       CHECK TOKEN
0044    0009 C1 03             CMP B #3       NUMBER?
0045    000B 26 22             BNE NOTNUM     NO
0046                           *
0047    000D 7D 0027           TST VALUE      NUMBER TOO BIG?
0048    0010 26 28             BNE BADNUM     YES, ERROR
0049                           *
0050    0012 96 28             LDA A VALUE+1  NUMBER TOO BIG?
0051    0014 81 02             CMP A #2       (3 DRIVES FOR SMOKE)
0052    0016 22 22             BHI BADNUM     YES, ERROR
0053                           *
0054    0018 CE 0063 R         LDX #PRMPT2    ISSUE SECOND PROMPT
0055    001B 8B 30             ADD A #$30     MAKE DRIVE NO. ASCII
0056    001D B7 0069 R         STA A DNUM
0057                           PRTMSG
0058 +  0020 3F                SWI
0059 +  0021 31                FCB 49
0060                           GTCMD          GET RESPONSE
```

```
0061 +  0022 3F               SWI
0062 +  0023 30               FCB 48
0063    0024 DE 20            LDX DESCRA
0064    0026 A6 00            LDA A 0,X
0065    0028 81 59            CMP A #'Y      WAS RESPONSE "YES"?
0066    002A 26 D4            BNE START      IF NOT, RETRY
0067                  *
0068    002C 7E 0073 R        JMP FORM2      IF SO, BEGIN FORMATTING
0069                  *
0070    002F DE 20    NOTNUM  LDX DESCRA     CHECK FOR "ESCAPE"
0071    0031 A6 00            LDA A 0,X
0072    0033 91 43            CMP A ES
0073    0035 26 03            BNE BADNUM     IF NOT, ERROR
0074                  *
0075                          INITDK         RE-INIT. DISK DRIVES
0076 +  0037 3F               SWI
0077 +  0038 33               FCB 51
0078    0039 39               RTS            IF SO, RETURN TO SYSTEM
0079                  *
0080    003A CE 0041 R BADNUM LDX #BADMSG    ERROR MESSAGE
0081                          PRTMSG
0082 +  003D 3F               SWI
0083 +  003E 31               FCB 49
0084    003F 20 BF            BRA START      RETRY
0085                  *
0086    0041 20       BADMSG  FCC ' BAD DRIVE NUMBER'
0087    0052 0D               FCB $0D
0088                  *
0089    0053 0A       PRMPT1  FCB $0A
0090    0054 44               FCC 'DRIVE NUMBER? '
0091    0062 04               FCB $04
0092                  *
0093    0063 44       PRMPT2  FCC 'DRIVE '
0094    0069 0001     DNUM    RMB 1
0095    006A 20               FCC ' READY? '
0096    0072 04               FCB $04
0098                  * FORMAT DISK HERE
0099                  *
0100    0073 7F 008A R FORM2  CLR TRACK      START AT TRACK 0
0101                  *
0102    0076 BD 0092 R FORM2A JSR TRKBLD     BUILD TRACK IMAGE
0103    0079 BD 0110 R        JSR TRKWRT     WRITE TRACK IMAGE TO DISK
0104    007C B6 008A R        LDA A TRACK
0105    007F 4C               INC A
0106    0080 B7 008A R        STA A TRACK    BUMP TRACK
0107    0083 81 23            CMP A #35      DONE?
0108    0085 26 EF            BNE FORM2A     LOOP UNTIL DONE
0109                  *
0110    0087 7E 0000 R        JMP START      BACK TO BEGINNING
0111                  *
0112    008A 0001     TRACK   RMB 1
0113    008B 0001     SECTOR  RMB 1
0114    008C 0002     SAVEX   RMB 2          DUMMY RCB (FOR "PRTERR")
0115    008E 46               FCC 'FMT'      'FMT' ERROR
0116    0091 0001     ERRCOD  RMB 1          DUMMY "FCBSTA"
0118                  * FORM A TRACK IMAGE IN MEMORY "TRKBUF" (3400 BYTES)
0119                  *
0120    0092 CE 0154 R TRKBLD LDX #TRKBUF    POINT TO BUFFER
0121    0095 86 FF            LDA A #$FF
0122    0097 C6 08            LDA B #8
0123    0099 8D 61            BSR PUTBYT     8-BYTE GAP
0124    009B 86 01            LDA A #1
0125    009D B7 008B R        STA A SECTOR   START OF SECTOR LOOP
0126                  *
0127                  * LOOP FOR SECTORS 1-18
0128                  *
0129    00A0 86 FF    SECLOP  LDA A #$FF
0130    00A2 C6 07            LDA B #7
0131    00A4 8D 56            BSR PUTBYT     7-BYTE GAP
0132    00A6 4F               CLR A
0133    00A7 C6 04            LDA B #4
0134    00A9 8D 51            BSR PUTBYT     4-BYTE SYNC
0135    00AB 86 FE            LDA A #$FE
0136    00AD A7 00            STA A 0,X      ID-ADDRESS MARK
0137    00AF 08               INX
0138    00B0 B6 008A R        LDA A TRACK
0139    00B3 A7 00            STA A 0,X      TRACK NO.
0140    00B5 08               INX
0141    00B6 6F 00            CLR 0,X        ZERO
0142    00B8 08               INX
0143    00B9 B6 008B R        LDA A SECTOR
0144    00BC A7 00            STA A 0,X      SECTOR NO.
0145    00BE 08               INX
0146    00BF 6F 00            CLR 0,X        LENGTH=128
0147    00C1 08               INX
0148    00C2 86 F7            LDA A #$F7
0149    00C4 A7 00            STA A 0,X      CRC
0150    00C6 08               INX
0151    00C7 86 FF            LDA A #$FF
0152    00C9 C6 0B            LDA B #11
0153    00CB 8D 2F            BSR PUTBYT     11-BYTE ID-GAP
0154    00CD 4F               CLR A
0155    00CE C6 06            LDA B #6
0156    00D0 8D 2A            BSR PUTBYT     6-BYTE SYNC
0157    00D2 86 FB            LDA A #$FB
0158    00D4 A7 00            STA A 0,X      DATA-ADDRESS MARK
0159    00D6 08               INX
0160    00D7 4F               CLR A
0161    00D8 C6 80            LDA B #128
0162    00DA 8D 20            BSR PUTBYT     128 BYTES OF DATA (0000)
0163    00DC 86 F7            LDA A #$F7
0164    00DE A7 00            STA A 0,X      CRC
0165    00E0 08               INX
0166    00E1 86 FF            LDA A #$FF
0167    00E3 A7 00            STA A 0,X      PAD
0168    00E5 08               INX
0169                  *
0170                  * END OF SECTOR DATA
0171                  *
0172    00E6 B6 008B R        LDA A SECTOR
0173    00E9 4C               INC A
0174    00EA B7 008B R        STA A SECTOR   BUMP SECTOR
0175    00ED 81 13            CMP A #19      DONE?
0176    00EF 26 AF            BNE SECLOP     LOOP THROUGH 18 SECTORS
0177                  *
0178                  * FINISH OUT TRACK WITH LONG GAP
0179                  *
0180    00F1 86 FF            LDA A #$FF
0181    00F3 C6 C8            LDA B #200
0182    00F5 8D 05            BSR PUTBYT
0183    00F7 C6 C8            LDA B #200
0184    00F9 8D 01            BSR PUTBYT
```

```
0185   00FB 39                RTS             DONE!
0186                  *
0187                  *
0188   00FC A7 00     PUTBYT  STA A 0,X       PUT BYTE INTO TRKBUF
0189   00FE 08                INX
0190   00FF 5A                DEC B           DONE?
0191   0100 26 FA             BNE PUTBYT      LOOP ON COUNT IN "B"
0192                  *
0193   0102 39                RTS
0195                  * WRITE TRACK IMAGE TO DISK DRIVE
0196                  * IMAGE IN "TRKBUF", DRIVE NO.  IN "VALUE+1"
0197                  *
0198                  * DISK ERRORS HANDLED HERE
0199                  *
0200   0103 F7 0091 R DSKERR  STA B ERRCOD
0201   0106 CE 008C R         LDX #SAVEX
0202                          PRTERR          ISSUE ERROR MESSAGE
0203 +  0109 3F               SWI
0204 +  010A 1E               FCB 30
0205   010B 31                INS
0206   010C 31                INS             CLEAN STACK (JSR TRKWRT)
0207   010D 7E 0000 R         JMP START       RETRY
0208                  *
0209                  *
0210   0110 96 28     TRKWRT  LDA A VALUE+1   GET DRIVE NUMBER
0211   0112 26 04             BNE SET1        NOT ZERO?
0212                  *
0213   0114 C6 08             LDA B #$08      DRIVE NO.  IN BFD-68 FORMAT
0214   0116 20 13             BRA SETD
0215                  *
0216   0118 81 01     SET1    CMP A #1        DRIVE 1?
0217   011A 26 04             BNE SET2        NO
0218                  *
0219   011C C6 10             LDA B #$10      DRIVE 1 IN BFD-FORMAT
0220   011E 20 0B             BRA SETD
0221                  *
0222   0120 81 02     SET2    CMP A #2        DRIVE 2?
0223   0122 26 04             BNE SET3        NO
0224                  *
0225   0124 C6 20             LDA B #$20      DRIVE 2 IN BFD-FORMAT
0226   0126 20 03             BRA SETD
0227                  *
0228   0128 7E 003A R SET3    JMP BADNUM      NOT 0,1,2 IS ERROR
0229                  *
0230   012B F7 A07B   SETD    STA B $A07B     SET DRIVE IN PLACE
0231   012E B6 008A R         LDA A TRACK
0232   0131 B7 A07C           STA A $A07C     SET TRACK IN PLACE
0233   0134 B6 008B R         LDA A SECTOR
0234   0137 B7 A07D           STA A $A07D     SET SECTOR IN PLACE
0235   013A CE 0154 R         LDX #TRKBUF
0236   013D FF A07E           STX $A07E       SET BUFFER POINTER
0237   0140 BD 82AB           JSR $82AB       SEEK TRACK
0238   0143 BD 8032           JSR $8032       WRITE TRACK
0239   0146 5D                TST B           ERROR?
0240   0147 26 BA             BNE DSKERR      YES
0241                  *
0242   0149 39                RTS
0243                  *
0244   014A 0154    R TRKBUF  EQU *+10        START OF TRACK IMAGE
0245                  *
0246                          END
```

```
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDXAB 2200 M
BADMSG 0041 R
BADNUM 003A R
BASEQU 2A2A M
BMEM   0033
BS     0039
CHAIN  243A M
CLASS  0026
CLOSE  2369 M
CMEM   0037
CMPC   231B M
CMWC   2572 M
CUCHAR 0023
DELETE 2420 M
DESCRA 0020
DESCRC 0022
DIV16  2524 M
DL     003A
DNUM   0069 R
DP     003B
DPCNT  003C
DSKERR 0103 R
DX     0040
EJ     0041
EMEM   0035
ERRCOD 0091 R
ES     0043
FCBCHN 0029
FCBDEF 2650 M
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
FORM2  0073 R
FORM2A 0076 R
FORMAT 0000 RN
FRETAB 002B
GETDR  23EC M
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
IOHDR  2335 M
LDP    0044
LDPCNT 0045
LOADB  246E M
LWD    0046
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NL     003E
NOTNUM 002F R
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
PRMPT1 0053 R
PRMPT2 0063 R
PRTERR 2454 M
PRTMSG 250A M
PS     0042
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTBYT 00FC R
PUTDR  2406 M
RC     0025
RCBDEF 258C M
READ   23B8 M
REWIND 2384 M
SAVEX  008C R
SECLOP 00A0 R
SECTOR 008B R
SET1   0118 R
SET2   0120 R
SET3   0128 R
SETD   012B R
START  0000 R
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
TABX   219C M
TB     003F
TRACK  008A R
TRKBLD 0092 R
TRKBUF 0154 R
TRKWRT 0110 R
TXAB   2183 M
VALUE  0027
WD     003D
WRITE  23D2 M
XABX   21B5 M
```

```
0001   0000 0000   N        NAM DSKDR
0002                *
0003   0000 0003   N        ENT . RDSEC
0004   0000 0043   N        ENT . WTSEC
0005   0000 0000   N        ENT @INTDK
0006               *
0007               *        SINGLE SECTOR READ AND WRITE ROUTINES
0008               *        FOR THE PERCOM DISK DRIVE SYSTEM
0009               *
0010               *        COPYRIGHT 1978 BY HEMENWAY ASSOCIATES INC
0011               *        BOSTON MASS.  02111
0012               *        ALL RIGHTS RESERVED
0013               *
0014               *
0015               * FCB  EQU'S
0016               *
0017   0000 0000   FCBEQT EQU 0          EQUIPMENT TABLE ADDRESS
0018   0000 0002   FCBGDT EQU 2          GENERIC DEVICE TYPE
0019   0000 0005   FCBSTA EQU 5          STATUS
0020   0000 0006   FCBDTT EQU 6          DATA TRANSFER TYPE
0021   0000 0007   FCBDBA EQU 7          DATA BUFFER ADDRESS
0022               *
0023   0000 0009   FCBDRV EQU 9          DRIVE NUMBER
0024   0000 000A   FCBTRK EQU 10         TRACK NUMBER
0025   0000 000B   FCBSCT EQU 11         SECTOR NUMBER
0026   0000 000C   FCBFWD EQU 12         FWD LINK TRACK/SECTOR
0027   0000 000E   FCBBAK EQU 14         BACK LINK TRACK/SECTOR
0028               *
0029               * BASE PAGE EQUATES
0030               *
0031   0000 0000   DRV     EQU 0          DESIRED TRACK (MS 2 BITS)
0032   0000 0001   TRK     EQU 1          DESIRED TRACK
0033   0000 0002   SCTR    EQU 2          DESIRED SECTOR
0034   0000 0003   BAKLNK  EQU 3          BACKWARD LINK
0035   0000 0005   FWDLNK  EQU 5          FORWARD LINK
0036   0000 0007   BYTCNT  EQU 7          SECTOR BYTE COUNT
0037   0000 0008   ADDRES  EQU 8          DATA ADDRESS VECTOR
0038   0000 0014   TA      EQU $14        CONTINUATION ADDRESS
0039   0000 0016   TW      EQU $16        ALTERNATE TARGET ADDRESS
0040               *
0041               * VECTORS INTO DISK DRIVERS
0042               *
0043   0000 C00C   RDSECX EQU $C00C       READ A SECTOR
0044   0000 C00F   WTSECX EQU $C00F       WRITE A SECTOR
0045   0000 C027   INITRK EQU $C027       INITIALIZE DRIVES
0046               *
0047               *
0048               * INITIALIZE DRIVES
0049               *
0050   0000 7E C027 @INTDK JMP INITRK
0051               *
0052               *
0053               *
0054               * READ A SINGLE SECTOR
0055               * A, B=FCBADR
0056               *
0057               . RDSEC TABX           X:=FCBADR
0058 +  0003 3F             SWI
0059 +  0004 03             FCB 3
0060   0005 EE 07           LDX FCBDBA, X
0061   0007 DF 16           STX TW         INIT DATA BUFFER ADDRESS
0062                        TABX           X:=FCBADR
0063 +  0009 3F             SWI
0064 +  000A 03             FCB 3
0065   000B 8D 21           BSR GETDTS     GET DRIVE, TRACK AND SECTOR
0066                        PSHX           SAVE FCBADR
0067 +  000D 3F             SWI
0068 +  000E 05             FCB 5
0069   000F BD C00C         JSR RDSECX     READ A SECTOR
0070                        PULX           RESTORE FCBADR
0071 +  0012 3F             SWI
0072 +  0013 06             FCB 6
0073   0014 24 14           BCC RDSEC1     OK
0074               *
0075               * RE-SET ERROR CODES TO CP/68 VALUES
0076               *
0077   0016 81 00           CMP A #0       0 BECOMES 10
0078   0018 26 04           BNE *+6
0079               *
0080   001A 86 0A           LDA A #10
0081   001C 20 0D           BRA RDSEC0
0082               *
0083   001E 81 01           CMP A #1       1 BECOMES 18
0084   0020 26 04           BNE *+6
0085               *
0086   0022 86 12           LDA A #18
0087   0024 20 05           BRA RDSEC0
0088               *
0089   0026 86 05           LDA A #5       OTHERS BECOME 5
0090   0028 20 01           BRA RDSEC0
0091               *
0092   002A 4F     RDSEC1 CLR A           NO ERRORS=0
0093               *
0094   002B A7 05  RDSEC0 STA A FCBSTA, X  SET ERROR CODE
0095   002D 39            RTS
0096               *
0097               *
0098               * INIT BASE PAGE DRIVE, TRACK AND SECTOR
0099               *
0099   002E A6 09  GETDTS LDA A FCBDRV, X       GET DRIVE #
0100   0030 4C            INC A                 OFFSET +1
0101   0031 48            ASL A                 SHIFT TO BITS 7, 6
0102   0032 48            ASL A
0103   0033 48            ASL A
0104   0034 48            ASL A
0105   0035 48            ASL A
0106   0036 48            ASL A
0107   0037 97 00         STA A DRV
0108   0039 A6 0A         LDA A FCBTRK, X       GET TRACK
0109   003B 97 01         STA A TRK
0110   003D A6 0B         LDA A FCBSCT, X       GET SECTOR
0111   003F 4A            DEC A                 OFFSET -1
0112   0040 97 02         STA A SCTR
0113   0042 39            RTS
0114               *
0115               *
0116               *
0117               * WRITE A SINGLE SECTOR
0118               * A, B=FCBADR
0119               *
0120               . WTSEC TABX           X:=FCBADR
0121 +  0043 3F             SWI
```

```
0122 +  0044 03                FCB 3
0123    0045 EE 07             LDX FCBDBA,X
0124    0047 DF 14             STX TA          INIT DATA BUFFER ADDRESS
0125                           TABX            X:=FCBADR
0126 +  0049 3F                SWI
0127 +  004A 03                FCB 3
0128    004B 8D E1             BSR GETDTS      GET DRIVE,TRACK AND SECTOR
0129    004D 7F 0007           CLR BYTCNT      SET FOR FULL SECTOR
0130                   *
0131    0050 86 FF             LDA A #$FF
0132    0052 97 08             STA A ADDRES
0133    0054 97 09             STA A ADDRES+1
0134                   *
0135    0056 A6 0E             LDA A FCBBAK,X
0136    0058 97 03             STA A BAKLNK
0137    005A 96 0F             LDA A FCBBAK+1
0138    005C 97 04             STA A BAKLNK+1
0139                   *
0140                           PSHX            SAVE FCBADR
0141 +  005E 3F                SWI
0142 +  005F 05                FCB 5
0143    0060 EE 0C             LDX FCBFWD,X    GET FWD LINK
0144    0062 DF 05             STX FWDLNK
0145                   *
0146    0064 BD C00F           JSR WTSECX      WRITE A SECTOR
0147                   *
0148                           PULX            X:=RCADR
0149 +  0067 3F                SWI
0150 +  0068 06                FCB 6
0151    0069 24 14             BCC WTSEC1      OK
0152                   *
0153                   * RE-SET ERROR CODES TO CP/68 VALUES
0154                   *
0155    006B 81 00             CMP A #0        0 BECOMES 10
0156    006D 26 04             BNE *+6
0157                   *
0158    006F 86 0A             LDA A #10
0159    0071 20 0D             BRA WTSEC0
0160                   *
0161    0073 81 01             CMP A #1        1 BECOME 18
0162    0075 26 04             BNE *+6
0163                   *
0164    0077 86 12             LDA A #18
0165    0079 20 05             BRA WTSEC0
0166                   *
0167    007B 86 05             LDA A #5        OTHERS BECOME 5
0168    007D 20 01             BRA WTSEC0
0169                   *
0170    007F 4F        WTSEC1  CLR A           NO ERRORS=0
0171                   *
0172    0080 A7 05     WTSEC0  STA A FCBSTA,X
0173    0082 39                RTS
0174                           END
```

```
.RDSEC 0003 RN
.WTSEC 0043 RN
@INIDK 0000 RN
ADDABX 2219 M
ADDAX  2232 M
ADDBX  224B M
ADDRES 0008
ADDXAB 2200 M
BAKLNK 0003
BASEQU 2A2A M
BYTCNT 0007
CHAIN  243A M
CLOSE  2369 M
CMPC   231B M
CMWC   2572 M
DELETE 2420 M
DIV16  2524 M
DRV    0000
DSKDR  0000 RN
FCBBAK 000E
FCBDBA 0007
FCBDEF 2650 M
FCBDRV 0009
FCBDTT 0006
FCBEQT 0000
FCBFWD 000C
FCBGDT 0002
FCBSCT 000B
FCBSTA 0005
FCBTRK 000A
FIBDEF 2940 M
FMTFCB 2488 M
FMTS   2558 M
FWDLNK 0005
GETDR  23EC M
GETDTS 002E R
GTCMD  24F0 M
INDEX  24BC M
INITDK 253E M
INITRK C027
IOHDR  2335 M
LOADR  246E M
MOVC   2301 M
MOVS   24A2 M
MUL16  22E7 M
MUL8   22CD M
NXTOK  24D6 M
OPEN   234F M
OPEND  239E M
PRTERR 2454 M
PRTMSG 250A M
PSHALL 2151 M
PSHX   21CE M
PULLAL 216A M
PULX   21E7 M
PUTDR  2406 M
RCBDEF 258C M
RDSEC0 002B R
RDSEC1 002A R
RDSECX C00C
READ   23B8 M
REWIND 2384 M
SCTR   0002
SUBABX 227F M
SUBAX  2299 M
SUBBX  22B3 M
SUBXAB 2265 M
TA     0014
TABX   219C M
TRK    0001
TW     0016
TXAB   2183 M
WRITE  23D2 M
WTSEC0 0080 R
WTSEC1 007F R
WTSECX C00F
XABX   21B5 M
```

```
0001   0000 0000    N        NAM INITER
0002                 *
0003                 * INITIALIZE A DISK FOR CP-68 OPERATING SYSTEM
0004                 *
0005                 * FOR PERCOM FLOPPY DISKS
0006                 *
0007                 * TRACK 0, SECTOR  1         HEADER OF FREE-SPACE LIST
0008                 * TRACK 0, SECTORS 2-10      DIRECTORY SPACE
0009                 * TRACKS 1-35                FREE-SPACE
0010                 *
0011                 * DISK ATTRIBUTES
0012                 *
0013   0000 0100     SECSIZ EQU 256        256 BYTES PER SECTOR
0014   0000 000A     TRKSIZ EQU 10         10 SECTORS PER TRACK
0015   0000 0022     DSKSIZ EQU 34         35 TRACKS ON DISK (LESS TRACK 0)
0016                 *
0017                 * FILE-CONTROL BLOCK ADDRESSES
0018                 *
0019   0000 0005     FCBSTA EQU 5          ERROR STATUS FLAG
0020   0000 0007     FCBDBA EQU 7          DATA BUFFER ADDRESS
0021   0000 0009     FCBDRV EQU 9          DRIVE NUMBER
0022   0000 000A     FCBTRK EQU 10         TRACK NUMBER
0023   0000 000B     FCBSCT EQU 11         SECTOR NUMBER
0024   0000 000C     FCBTLK EQU 12         TRACK LINK POINTER
0025   0000 000D     FCBSLK EQU 13         SECTOR LINK POINTER
0026                 *
0027   0000 0002     FCBSPC RMB 2          FILE-CONTROL BLOCK
0028   0002 44              FCC 'DSK'      DISK
0029   0005 0001            RMB 1
0030   0006 FF              FCB $FF        OUTPUT
0031   0007 0023            RMB 35
0032                 *
0033   002A 0100     BUFFER RMB SECSIZ     SECTOR BUFFER
0034                 *
0035                 * COMMAND-LINE INTERPRETER BASE-PAGE LOCATIONS
0036                 *
0037   012A 0020     DESCRA EQU $20        ADDRESS OF TOKEN
0038   012A 0027     VALUE  EQU $27        VALUE OF NUMERIC TOKEN
0039                 *
0040   012A 49       PROMPT FCC 'INIT. DISK IN DRIVE '
0041   013E 0001     DRVNO  RMB 1
0042   013F 20              FCC ' ? '
0043   0142 04              FCB $04
0044                 *
0045   0143 0143    N        ENT . INITR    ENTRY POINT FROM CLI
0046                 *
0047   0143 96 28    . INITR LDA A VALUE+1 GET DRIVE NUMBER
0048   0145 84 03            AND A #$03    LIMIT RANGE (ICOM PERMITS 4 DRIVES)
0049   0147 CE 0000 R        LDX #FCBSPC   POINT TO FCB
0050   014A A7 09            STA A FCBDRV,X
0051   014C 8B 30            ADD A #$30    MAKE DRIVE NUMBER ASCII
0052   014E B7 013E R        STA A DRVNO   PUT IN PROMPT LINE
0053   0151 CE 012A R        LDX #PROMPT
0054                         PRTMSG        OUTPUT PROMPT
0055 +  0154 3F               SWI
0056 +  0155 31               FCB 49
0057                         GTCMD         GET USER RESPONSE
0058 +  0156 3F               SWI
0059 +  0157 30               FCB 48
0060   0158 DE 20            LDX DESCRA
0061   015A A6 00            LDA A 0,X     GET FIRST CHAR. OF RESPONSE
0062   015C 81 59            CMP A #'Y     WAS IT 'YES'?
0063   015E 27 01            BEQ INITR2    IF SO, CONTINUE
0064                 *
0065   0160 39               RTS           IF NOT, QUIT
0066                 *
0067   0161 CE 0000 R INITR2 LDX #FCBSPC   POINT TO FCB
0068   0164 6F 0A            CLR FCBTRK,X  TRACK=0
0069   0166 86 01            LDA A #1
0070   0168 A7 0B            STA A FCBSCT,X  SECTOR=1
0071                 *
0072                 * INITIALIZE HEAD OF FREE-SPACE BLOCK
0073                 *
0074                 * ALL ZERO EXCEPT FOR LAST TWO BYTES=TRACK 1, SECTOR 1
0075                 *
0076                         TXAB
0077 +  016A 3F               SWI
0078 +  016B 02               FCB 2
0079   016C CE 002A R        LDX #BUFFER
0080                         XABX
0081 +  016F 3F               SWI
0082 +  0170 04               FCB 4
0083   0171 A7 07            STA A FCBDBA,X
0084   0173 E7 08            STA B FCBDBA+1,X
0085                         PSHX
0086 +  0175 3F               SWI
0087 +  0176 05               FCB 5
0088                 *
0089                 * CLEAR OUT BUFFER EXCEPT FOR LAST 2 BYTES
0090                 *
0091   0177 CE 002A R        LDX #BUFFER
0092   017A C6 FE            LDA B #SECSIZ-2
0093   017C 4F               CLR A
0094   017D A7 00    INITR3  STA A 0,X
0095   017F 08               INX
0096   0180 5A               DEC B
0097   0181 26 FA            BNE INITR3
0098                 *
0099   0183 86 01            LDA A #1      TRACK, SECTOR=1
0100   0185 A7 00            STA A 0,X
0101   0187 A7 01            STA A 1,X
0102                         PULX
0103 +  0189 3F               SWI
0104 +  018A 06               FCB 6
0105   018B 8D 7E            BSR WRTBLK    WRITE BLOCK 1
0106   018D 6D 05            TST FCBSTA,X  CHECK FOR DISK ERROR
0107   018F 27 04            BEQ *+6       OK
0108                 *
0109   0191 20 4F            BRA INITQ     FATAL DISK ERROR, QUIT
0110                 *
0111   0193 20 76    @WRTBL  BRA WRTBLK    OUT OF RANGE "BSR WRTBLK"
0112   0195 6C 0B            INC FCBSCT,X  SECTOR=2
0113   0197 7F 0128 R        CLR BUFFER+SECSIZ-2
0114   019A 7F 0129 R        CLR BUFFER+SECSIZ-1
0115                 *
0116                 * INITIALIZE DIRECTORY TO ZERO
0117                 *
0118   019D 8D 6C    INITR4  BSR WRTBLK    WRITE DIRECTORY BLOCK
0119   019F 6D 05            TST FCBSTA,X  CHECK FOR DISK ERROR
0120   01A1 27 02            BEQ *+4       OK
0121                 *
```

```
0122   01A3 20 3D            BRA INITQ     FATAL DISK ERROR, QUIT
0123                 *
0124   01A5 A6 0B            LDA A FCBSCT,X
0125   01A7 4C               INC A         NEXT SECTOR
0126   01A8 81 0B            CMP A #TRKSIZ+1 DONE WITH TRACK?
0127   01AA 27 04            BEQ INITR5    YES
0128                 *
0129   01AC A7 0B            STA A FCBSCT,X
0130   01AE 20 ED            BRA INITR4    NO, CONTINUE WRITING
0131                 *
0132   01B0 86 01    INITR5  LDA A #1
0133   01B2 A7 0B            STA A FCBSCT,X  SECTOR=1
0134   01B4 A7 0A            STA A FCBTRK,X  TRACK=1
0135   01B6 16               TAB
0136                 *
0137                 * INITIALIZE REST OF DISK (FREE-SPACE)
0138                 *
0139                 *       X=FCB ADDRESS
0140                 *       A=TRACK NUMBER
0141                 *       B=SECTOR NUMBER
0142                 *
0143   01B7 5C       INITR6  INC B         MAKE SECTOR LINKAGE
0144   01B8 C1 0B            CMP B #TRKSIZ+1 END OF TRACK?
0145   01BA 26 09            BNE INITR7    NO
0146                 *
0147   01BC C6 01            LDA B #1      YES, SECTOR=1
0148   01BE 4C               INC A         NEXT TRACK
0149   01BF 81 23            CMP A #DSKSIZ+1 END OF DISK?
0150   01C1 26 02            BNE INITR7    NO
0151                 *
0152   01C3 4F               CLR A         LAST SECTOR POINTS TO 0,0
0153   01C4 5F               CLR B
0154                 *
0155   01C5 B7 002A R INITR7 STA A BUFFER  TRACK LINK
0156   01C8 37               PSH B         SAVE LSEC
0157   01C9 8D 33            BSR GETSC     GET PSEC
0158   01CB F7 002B R        STA B BUFFER+1 SECTOR LINK
0159   01CE 33               PUL B         RESTORE LSEC
0160   01CF 8D 3A            BSR WRTBLK    WRITE SECTOR
0161   01D1 4D               TST A         DONE? (=0)
0162   01D2 26 04            BNE INITR8    NO
0163                 *
0164   01D4 5D               TST B         DONE? (=0)
0165   01D5 26 01            BNE INITR8    NO
0166                 *
0167   01D7 39               RTS           YES, DONE!!!
0168                 *
0169   01D8 A7 0A    INITR8  STA A FCBTRK,X
0170   01DA 37               PSH B         SAVE LSEC
0171   01DB 8D 21            BSR GETSC     GET PSEC
0172   01DD E7 0B            STA B FCBSCT,X
0173   01DF 33               PUL B         GET LSEC
0174   01E0 20 D5            BRA INITR6    KEEP WRITING
0175                 *
0176                 *
0177                 * FATAL ERROR MESSAGE
0178                 *
0179   01E2 CE 01E8 R INITQ  LDX #QMSG     OUTPUT ERROR MESSAGE
0180                         PRTMSG
0181 +  01E5 3F               SWI
0182 +  01E6 31               FCB 49
0183   01E7 39               RTS           RETURN TO CLI
0184                 *
0185   01E8 49       QMSG    FCC 'INITIALIZATION FAILED'
0186   01FD 0D               FCB $0D
0187                 *
0188                 * CONVERT LSEC TO PSEC
0189                 * LSEC IN B-REG
0190                 *
0191                 GETSC   PSHX          SAVE X-REGISTER
0192 +  01FE 3F               SWI
0193 +  01FF 05               FCB 5
0194   0200 CE 027D R        LDX #TBL      POINT TO LOGICAL/PHYSICAL TABLE
0195                         ADDBX         ADD LOGICAL OFFSET
0196 +  0203 3F               SWI
0197 +  0204 0A               FCB 10
0198   0205 09               DEX           SECTOR STARTS AT 1
0199   0206 E6 00            LDA B 0,X     GET PSEC
0200                         PULX          RESTORE X-REG
0201 +  0208 3F               SWI
0202 +  0209 06               FCB 6
0203   020A 39               RTS
0204                 *
0205                 * WRITE A SECTOR WITH ERROR CHECKING
0206                 *
0207   020B 36       WRTBLK  PSH A         SAVE 'A'
0208   020C 6F 05            CLR FCBSTA,X  CLEAR ERROR FLAG
0209                         IOHDR         ISSUE I/O REQUEST
0210 +  020E 3F               SWI
0211 +  020F 13               FCB 19
0212   0210 6D 05            TST FCBSTA,X  ERROR?
0213   0212 26 02            BNE WRTERR    YES
0214                 *
0215   0214 32               PUL A         RESTORE 'A'
0216   0215 39               RTS
0217                 *
0218   0216 16       WRTERR  TAB
0219   0217 8D 54            BSR OUTHL     CONVERT LEFT DIGIT
0220   0219 B7 0253 R        STA A ERTYPE
0221   021C 17               TBA
0222   021D 8D 52            BSR OUTHR     CONVERT RIGHT DIGIT
0223   021F B7 0254 R        STA A ERTYPE+1
0224                         PSHX          SAVE X
0225 +  0222 3F               SWI
0226 +  0223 05               FCB 5
0227   0224 A6 0B            LDA A FCBSCT,X
0228   0226 8D 45            BSR OUTHL     MAKE SECTOR NO. HEX
0229   0228 B7 0260 R        STA A SECT
0230   022B A6 0B            LDA A FCBSCT,X
0231   022D 8D 42            BSR OUTHR
0232   022F B7 0261 R        STA A SECT+1
0233   0232 A6 0A            LDA A FCBTRK,X
0234   0234 8D 37            BSR OUTHL     MAKE TRACK NO. HEX
0235   0236 B7 026A R        STA A TRACK
0236   0239 A6 0A            LDA A FCBTRK,X
0237   023B 8D 34            BSR OUTHR
0238   023D B7 026B R        STA A TRACK+1
0239   0240 CE 0248 R        LDX #DERROR
0240                         PRTMSG        PRINT ERROR MESSAGE
0241 +  0243 3F               SWI
0242 +  0244 31               FCB 49
0243   0245 3F               SWI           CALL CP/68
```

```
0244   0246 1F             FCB 31          "WARMSTART"
0245   0247 39             RTS             QUIT
0246                *
0247   0248 44      DERROR FCC 'DISK ERROR: '
0248   0253 0002    ERTYPE RMB 2
0249   0255 20             FCC ' AT SECTOR '
0250   0260 0002    SECT   RMB 2
0251   0262 2C             FCC ', TRACK '
0252   026A 0002    TRACK  RMB 2
0253   026C 0D             FCB $0D
0254                *
0255                * CONVERT BINARY TO HEX-ASCII HERE
0256                *
0257   026D 44      OUTHL  LSR A           SHIFT RIGHT
0258   026E 44             LSR A
0259   026F 44             LSR A
0260   0270 44             LSR A
0261                *
0262   0271 84 0F   OUTHR  AND A #$0F      GET NIBBLE
0263   0273 8B 30          ADD A #$30      MAKE ASCII
0264   0275 81 39          CMP A #$39      >9?
0265   0277 23 02          BLS *+4         NO
0266                *
0267   0279 8B 07          ADD A #$7       YES
0268                *
0269   027B 39             RTS
0270                *
0271                *
0272                * LOGICAL/PHYSICAL SECTOR TABLE
0273                *
0274   027C 00             FCB 00
0275                *
0276   027D 01      TBL    FCB $1
0277   027E 05             FCB $5
0278   027F 09             FCB $9
0279   0280 03             FCB $3
0280   0281 07             FCB $7
0281   0282 02             FCB $2
0282   0283 06             FCB $6
0283   0284 0A             FCB $A
0284   0285 04             FCB $4
0285   0286 08             FCB $8
0286                       END
```

```
. INITR 0143 RN
@WRTBL  0193 R
ADDABX  2219 M
ADDAX   2232 M
ADDBX   224B M
ADDXAB  2200 M
BASEQU  2A2A M
BUFFER  002A R
CHAIN   243A M
CLOSE   2369 M
CMPC    231B M
CMWC    2572 M
DELETE  2420 M
DERROR  0248 R
DESCRA  0020
DIV16   2524 M
DRVNO   013E R
DSKSIZ  0022
ERTYPE  0253 R
FCBDBA  0007
FCBDEF  2650 M
FCBDRV  0009
FCBSCT  000B
FCBSLK  000D
FCBSPC  0000 R
FCBSTA  0005
FCBTLK  000C
FCBTRK  000A
FIBDEF  2940 M
FMTFCB  2488 M
FMTS    2558 M
GETDR   23EC M
GETSC   01FE R
GTCMD   24F0 M
INDEX   24BC M
INITDK  253E M
INITER  0000 RN
INITQ   01E2 R
INITR2  0161 R
INITR3  017D R
INITR4  019D R
INITR5  01B0 R
INITR6  01B7 R
INITR7  01C5 R
INITR8  01D8 R
IOHDR   2335 M
LOADB   246E M
MOVC    2301 M
MOVS    24A2 M
MUL16   22E7 M
MUL8    22CD M
NXTOK   24D6 M
OPEN    234F M
OPEND   239E M
OUTHL   026D R
OUTHR   0271 R
PROMPT  012A R
PRTERR  2454 M
PRTMSG  250A M
PSHALL  2151 M
PSHX    21CE M
PULLAL  216A M
PULX    21E7 M
PUTDR   2406 M
QMSG    01E8 R
RCBDEF  258C M
READ    23B8 M
REWIND  2384 M
SECSIZ  0100
SECT    0260 R
SUBABX  227F M
SUBAX   2299 M
SUBBX   22B3 M
SUBXAB  2265 M
TABX    219C M
TBL     027D R
TRACK   026A R
TRKSIZ  000A
TXAB    2183 M
VALUE   0027
WRITE   23D2 M
WRTBLK  020B R
WRTERR  0216 R
XABX    21B5 M
```

```
0001   0000 0000     N        NAM BOOT
0002                  *
0003                  * PERCOM CP/68 BOOTSTRAP PROGRAM
0004                  * ASSUMES SYSTEM FILE LINKED AS FOLLOWS:
0005                  *
0006                  *      TRACK 0, SECTOR 1, BYTE 250-FIRST TRACK
0007                  *                             251-FIRST SECTOR
0008                  *                             252-LAST TRACK
0009                  *                             253-LAST SECTOR
0010                  *                             254,5 FREE-SPACE HEADER
0011                  *
0012                  * BOOTS SYSTEM FROM DRIVE 0:
0013                  *
0014                  * DEFINE DISK-DRIVE INTERFACE ADDRESSING
0015                  *
0016   0000 C027      INITRK EQU $C027
0017   0000 C00C      RDSECX EQU $C00C
0018                  *
0019   0000 0000      DRV    EQU $0000
0020   0000 0001      TRK    EQU $0001
0021   0000 0002      SCTR   EQU $0002
0022   0000 0016      TW     EQU $0016
0023   0000 0100      SECSIZ EQU 256
0024                  *
0025                  * NOTE: ALL VARIABLES IN COMMON, CODE IS ROM-ABLE
0026                  *
0027   0000 0000     C        CMN STACK,16
0028   0000 0010     C        CMN BUFFER,SECSIZ
0029   0000 0110     C        CMN FTS,2
0030   0000 0112     C        CMN LTS,2
0031   0000 0114     C        CMN PTS,2
0032   0000 0116     C        CMN INDEX,2
0033   0000 0118     C        CMN SAVEX,2
0034   0000 011A     C        CMN ADDRES,2
0035   0000 011C     C        CMN FCNT,1
0036                  *
0037                  * ERROR JUMP VECTOR
0038                  *
0039   0000 E113      ERROR  EQU $E113
0040                  *
0041                  * BEGIN BOOT HERE
0042                  *
0043   0000 8E 000F C START  LDS #STACK+15 INIT. STACK POINTER
0044   0003 BD C027          JSR INITRK    INIT. DRIVES
0045                  *
0046                  * NOW GET SYSTEM LINK INFORMATION
0047                  *
0048   0006 86 01            LDA A #1
0049   0008 C6 00            LDA B #0      TRACK 0, SECTOR 1
0050   000A CE 0010 C        LDX #BUFFER
0051   000D BD 00B5 R        JSR RDSEC     READ LINK SECTOR
0052   0010 CE 0010 C        LDX #BUFFER
0053   0013 A6 FA            LDA A SECSIZ-6,X   GET FIRST T/S
0054   0015 E6 FB            LDA B SECSIZ-5,X
0055   0017 B7 0110 C        STA A FTS
0056   001A F7 0111 C        STA B FTS+1
0057   001D A6 FC            LDA A SECSIZ-4,X   GET LAST T/S
0058   001F E6 FE            LDA B SECSIZ-2,X
0059   0021 B7 0112 C        STA A LTS
0060   0024 F7 0113 C        STA B LTS+1
0061   0027 CE 0014 C        LDX #BUFFER+4
0062   002A FF 0116 C        STX INDEX     INIT. BUFFER INDEX
0063   002D B6 0111 C        LDA A FTS+1
0064   0030 F6 0110 C        LDA B FTS
0065   0033 B7 0115 C        STA A PTS+1   INIT. PRESENT T/S
0066   0036 F7 0114 C        STA B PTS
0067   0039 CE 0010 C        LDX #BUFFER
0068   003C BD 00B5 R        JSR RDSEC     READ FIRST SECTOR
0069                  *
0070                  * NOW LOAD SYSTEM FILE INTO MEMORY
0071                  *
0072   003F 8D 3A     BOOT1  BSR GETBYT    GET A DATA BYTE FROM FILE
0073   0041 81 16            CMP A #$16    TRANSFER-ADDRESS?
0074   0043 26 0C            BNE BOOT2     NO
0075                  *
0076   0045 8D 34            BSR GETBYT
0077   0047 B7 011A C        STA A ADDRES  GET TRANSFER ADDRESS
0078   004A 8D 2F            BSR GETBYT
0079   004C B7 011B C        STA A ADDRES+1
0080   004F 20 EE            BRA BOOT1     GET NEW DATA FRAME
0081                  *
0082   0051 81 02     BOOT2  CMP A #$02    DATA FRAME?
0083   0053 26 21            BNE BOOT4     NO
0084                  *
0085   0055 8D 24            BSR GETBYT
0086   0057 B7 0118 C        STA A SAVEX   GET ADDRESS
0087   005A 8D 1F            BSR GETBYT
0088   005C B7 0119 C        STA A SAVEX+1
0089   005F 8D 1A            BSR GETBYT
0090   0061 B7 011C C        STA A FCNT    GET FRAME COUNTER
0091                  *
0092   0064 8D 15     BOOT3  BSR GETBYT    GET DATA BYTE
0093   0066 FE 0118 C        LDX SAVEX
0094   0069 A7 00            STA A 0,X     STORE BYTE
0095   006B 08               INX
0096   006C FF 0118 C        STX SAVEX
0097   006F 7A 011C C        DEC FCNT      COUNT DOWN
0098   0072 26 F0            BNE BOOT3
0099                  *
0100   0074 20 C9            BRA BOOT1     GET NEW DATA FRAME
0101                  *
0102   0076 FE 011A C BOOT4  LDX ADDRES    GET TRANSFER ADDRESS
0103   0079 6E 00            JMP 0,X       GO THERE
0104                  *
0106                  * READ A DATA BYTE FROM SYSTEM FILE
0107                  * RETURN BYTE IN 'A' REGISTER
0108                  *
0109   007B FE 0116 C GETBYT LDX INDEX
0110   007E 8C 0110 C        CPX #BUFFER+256      NEED NEW SECTOR?
0111   0081 27 07            BEQ GETSEC           YES
0112                  *
0113   0083 A6 00            LDA A 0,X            GET BYTE
0114   0085 08               INX
0115   0086 FF 0116 C        STX INDEX            MOVE POINTER
0116   0089 39               RTS
0117                  *
0118   008A F6 0114 C GETSEC LDA B PTS
0119   008D B6 0115 C        LDA A PTS+1          CHECK FOR LAST SECTOR
0120   0090 B1 0113 C        CMP A LTS+1
0121   0093 26 07            BNE GETS2            NOT LAST
0122                  *
```

```
0123   0095 F1 0112 C        CMP B LTS
0124   0098 26 02            BNE GETS2           NOT LAST
0125                  *
0126   009A 20 DA            BRA BOOT4           EOF-GO TO TRANSFER ADDRESS
0127                  *
0128   009C CE 0010 C GETS2  LDX #BUFFER
0129   009F E6 00            LDA B 0,X           GET FORWARD T/S LINK
0130   00A1 A6 01            LDA A 1,X
0131   00A3 F7 0114 C        STA B PTS           UPDATE PRESENT T/S
0132   00A6 B7 0115 C        STA A PTS+1
0133   00A9 8D 0A            BSR RDSEC           READ NEW SECTOR
0134   00AB CE 0014 C        LDX #BUFFER+4
0135   00AE A6 00            LDA A 0,X           GET DATA BYTE
0136   00B0 08               INX
0137   00B1 FF 0116 C        STX INDEX           RE-INIT. INDEX
0138   00B4 39               RTS
0139                  *
0141                  * SINGLE-SECTOR READ ROUTINE
0142                  *
0143                  *     DRIVE=0
0144                  *     TRACK='B'
0145                  *     SECTOR='A'
0146                  *     BUFFER='X'
0147                  *
0148   00B5 4A        RDSEC  DEC A               OFFSET OF SECTOR=-1
0149   00B6 97 02            STA A SCTR
0150   00B8 D7 01            STA B TRK
0151   00BA DF 16            STX TW              SAVE BUFFER ADDRESS
0152   00BC 86 40            LDA A #$40          INIT. DRIVE
0153   00BE 97 00            STA A DRV
0154   00C0 BD C00C          JSR RDSECX
0155   00C3 24 03            BCC *+5             ERROR?
0156                  *
0157   00C5 7E E113          JMP ERROR           YES
0158                  *
0159   00C8 39               RTS                 NO
0160                  *
0161                  END
```

```
ADDRES 011A C
BOOT   0000 RN
BOOT1  003F R
BOOT2  0051 R
BOOT3  0064 R
BOOT4  0076 R
BUFFER 0010 C
DRV    0000
ERROR  E113
FCNT   011C C
FTS    0110 C
GETBYT 007B R
GETS2  009C R
GETSEC 008A R
INDEX  0116 C
INITRK C027
LTS    0112 C
PTS    0114 C
RDSEC  00B5 R
RDSECX C00C
SAVEX  0118 C
SCTR   0002
SECSIZ 0100
STACK  0000 C
START  0000 R
TRK    0001
TW     0016
```

Index

A

ADABX 33
ADDAX 33
ADDBX 33
ADXAB 32
ASSIGN 3

B

BINFRM 77
BOOT 4
BUILD 95, 96

C

CHAIN 43, 66
CHRTAB 58, 62
CLI 58
CLOSE 41, 69
CMDTAB 58
CMPC 34
CMWC 39
CONRCB 59
CREATE 85
CVDB 61, 62
CVHB 62

D

DELETE 4, 42, 67, 95
DESCRA 62
DESCRC 62
DEVNAM 60
DEVTAB 75
DIRCMD 65
DIRECTORY 5, 40
DISPATCH 54
DIV16 35
DLKUP 75
DRIVER 80
DSCAN 61
DSKSIZ 100
DTDCPY 77

E

ENLARGE 95, 96
EQTAB 54
EXIT 6
EXPAND 87

F

FCBPOS 84
FCBRCD 84
FCBRNM 83
FCBRTB 84
FCBRSZ 84
FILCPY 77
FILENAMES 2
FMTFCB 38, 67
FMTS 39, 54

G

GCHRTB 62
GETBYT 72
GETDR 40
GETSC 74
GTCMD 38, 59

H

HEXFRM 77
HSCAN 62

I

INCON 56
INDEX 35
INICMD 65
INITDK 42, 59, 70
INITIALIZE 6, 73
INLIN 55
INRDR 56
IOHDR 37, 55

Index

J

JMPCMD 63
JUMP 7

L

LINK 7, 74
LOAD 7
LOADB 43, 64
LOCATE 95, 96
LODCMD 59, 64

M

MOVC 34
MOVS 35
MUL8 33
MUL16 34

N

NSCAN 61
NULL 55
NXTOK 35, 60

O

OPEN 41, 68
OPEND 40
OTLIN 56
OTPCH 57
OUTCON 56
OUTLPT 57
OUTPCH 57

P

PDSRCH 55, 71
PDTAB 54
PIP 8, 75
PLACE 95, 96
POSITION 87
PRTERR 38, 60
PRTMSG 38, 59
PSHAL 32
PULAL 32
PULX 32
PUTDR 40

R

RCLOSE 86
RDRIN 57
READ 42, 69
RENAME 10
RENCMD 64
REWD 41, 70
RDSEC 70, 73
ROPEN 86
RWRITE 87

S

SAVE 10
SAVCMD 64
SBABX 33
SBXAB 33
SECSIZ 99
SECURITY 11, 78
SEMPTY 11
SET 11, 78
SFILE 66
STATUS 13, 78
SUBAX 33
SUBBX 33
SUBCMD 63
SUBFCB 60
SUBFLG 60
SUBMIT 13

T

TABX 32
TRKBLD 80
TRKSIZ 100
TRKWLT 81
TXAB 32

U

USR1-USR11 43

V

VALUE 61

W

WARM3 59
WARMST 42, 59
WILDCARD 3
WRITE 42, 70
WRTBLK 74
WTSEC 71

X

XABX 32